

DEEP SEQUENTIAL AND STRUCTURAL NEURAL MODELS OF COMPOSITIONALITY

A Dissertation

Presented to the Faculty of the Graduate School
of Cornell University

in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by

Ozan İrsoy

January 2017

© 2017 Ozan İrsoy
ALL RIGHTS RESERVED

DEEP SEQUENTIAL AND STRUCTURAL NEURAL MODELS OF COMPOSITIONALITY

Ozan İrsoy, Ph.D.

Cornell University 2017

Recent advances in deep learning have provided fruitful applications for natural language processing (NLP) tasks. One key advance was the invention of word vectors, representing every word in a dense, low-dimensional vector space. Even though word vectors provide very strong results for word level NLP tasks, producing appropriate representation for phrases and sentences is still an open research problem.

In this dissertation, we focus on compositional approaches to representation learning. In particular, we employ the notions of compositionality in which the sequence or structure information is utilized, via recurrent or recursive neural networks. We investigate the effectiveness of such approaches for specific natural language understanding tasks including opinion mining and sentiment analysis, and extend some of the approaches to provide better representation hierarchies. In particular, we propose two novel variants: bidirectional recursive neural networks, which are capable of producing context-dependent structural representations and deep recursive neural networks, which provide representation hierarchies in the structural setting. Additionally, we qualitatively investigate such models, and describe how they relate to alternative compositional approaches. Finally, we discuss challenges in interpretation and understanding of compositional neural models, propose simple tools for visualization, and perform exploratory analyses over features learned by such a model.

BIOGRAPHICAL SKETCH

Ozan İrsoy was born in İzmir, Turkey. He pursued his Bachelor's degrees in Mathematics and Computer Engineering at Boğaziçi University. During his undergraduate studies, he developed an interest in machine learning. His early work on decision trees that started as part of his Bachelor's Thesis turned into several novel decision tree methods. Upon graduation, he has won several awards including Boğaziçi University Department of Computer Engineering and the School of Engineering Valedictorian, Double Major Graduate Award, and Best Bachelor's Thesis Award.

After completion of the Bachelor's degrees, he joined to Cornell University to pursue a doctorate degree in Computer Science. During his PhD, his research lied at the intersection of representation learning and natural language processing. He worked on learning (possibly deep) representations for compositionality in language, with applications to sentiment analysis, opinion mining and question answering. He has completed internships at Google and MetaMind applying neural networks to natural language understanding and speech recognition problems.

To Gizem, who made all of this possible. Look at where we are!

ACKNOWLEDGEMENTS

I am extremely thankful to my advisor Claire Cardie for her infinite guidance, support, mentorship, suggestions, ideas and all the help she provided. I am also very grateful to my committee members Robert Kleinberg and Dawn Woodard for their insightful input and feedback to my research.

I would like to thank my mentors who inspired and helped me during my internships, Andrew Senior and Hasim Sak from Google and Richard Socher from MetaMind. Special thanks to Volkan Cirik, Ainur Yessenalina, and Bishan Yang for helping with some of the experiments in this thesis, and Mohit Iyyer for insightful discussions.

Many thanks to Moontae Lee, Vlad Niculae, Arzoo Katiyar, Bishan Yang, Lu Wang, Jon Park, Adith Swaminathan, Jason Yosinski, Lillian Lee, David Mimno, and all the members of the NLP Seminar and MLDG for their helpful feedback.

I would like to thank my collaborator and Bachelor's thesis advisor Ethem Alpaydin for introducing me to machine learning and artificial intelligence.

I thank my beautiful wife Gizem, for being the source of love, happiness, joy and strength.

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
List of Tables	ix
List of Figures	x
Notation	xii
1 Introduction	1
1.1 Overview	1
1.1.1 Distributed Word Representations	2
1.1.2 Composition of Word Vectors	4
1.1.3 Artificial Neural Networks	5
1.2 Contributions	7
1.3 Roadmap	8
2 Background and Related Work	10
2.1 Representation Learning	10
2.2 Learning Word Representations	11
2.3 Learning Representations for Phrases and Sentences	13
2.3.1 Orderless Composition	14
2.3.2 Sequential Composition	15
2.3.3 Structural Composition	18
3 Opinion Mining with Deep Recurrent Neural Networks	20
3.1 Related Work	22
3.2 Sequential Neural Models for Opinion Mining	23
3.2.1 Recurrent Neural Networks	23
3.2.2 Bidirectionality	25
3.2.3 Depth in Space	27
3.3 Experiments	29
3.3.1 Hypotheses	29
3.3.2 Experimental Setting	29
3.3.3 Results and Discussion	32
3.4 Chapter Summary	38
4 Modeling Compositionality with Multiplicative Recurrent Neural Networks	39
4.1 Related Work	41
4.2 Composition with Matrix-Space Models	43
4.3 Composition with Multiplicative Recurrent Neural Networks	44
4.3.1 Ordinal regression with neural networks	45

4.3.2	Relationship to matrix-space model	46
4.4	Experiments	49
4.4.1	Experimental Setting	49
4.4.2	Results and Discussion	50
4.5	Chapter Summary	53
5	Bidirectional Recursive Neural Networks for Token-Level Labeling with Structure	55
5.1	Related Work	57
5.2	Structural Composition with Recursive Neural Networks	58
5.3	Bidirectional Recursive Neural Networks	60
5.3.1	Incorporating Sequential Context	62
5.4	Experiments	63
5.4.1	Experimental Setting	63
5.4.2	Results and Discussion	65
5.5	Chapter Summary	68
6	Deep Recursive Neural Networks for Compositionality in Language	70
6.1	Related Work	72
6.2	Deep Recursive Composition	73
6.2.1	Untying Leaves and Internals	74
6.2.2	Deep Recursive Neural Networks	75
6.3	Experiments	77
6.3.1	Experimental Setting	77
6.3.2	Results and Discussion	80
6.4	Chapter Summary	85
7	Towards Understanding Neural Language Learners	87
7.1	Related Work	88
7.2	Identifying Challenges	90
7.3	Visualizing Hidden Features	93
7.3.1	First Order Saliency and Expected Deviation	95
7.3.2	Cross-Feature First Order Saliency	97
7.3.3	Immediate Temporal Differences	98
7.3.4	Fine-grained Activation Statistics	99
7.4	Experiments	100
7.4.1	Experimental Setting	100
7.4.2	Results and Discussion	102
7.5	Chapter Summary	115
8	Conclusion and Future Work	119
8.1	Summary of Contributions	119
8.2	Future Work	120

A	Additional Visualizations of Model Activations	123
A.1	Cross-feature Saliency	123
A.2	Generic Sentiment Features	124
A.3	Decorrelated Features	125

LIST OF TABLES

3.1	An example sentence with opinion expression labels	21
3.2	Experimental evaluation of recurrent networks for DSE extraction	32
3.3	Experimental evaluation of recurrent networks for ESE extraction	33
3.4	Comparison of Deep recurrent networks to state-of-the-art (semi)CRF baselines for DSE and ESE detection	33
4.1	Average ranking losses (MPQA)	52
4.2	Average accuracies (SSTB)	52
5.1	An example sentence with opinion expression, holder, and target labels	57
5.2	Experimental results for DSE detection	65
5.3	Experimental results for ESE detection	65
5.4	Experimental results for joint HOLDER+DSE+TARGET detection	65
6.1	Results for RSVs. ℓ and $ h $ denote the depth and width of the networks, respectively.	80
6.2	Results for previous work and our best model (dRSV).	81
6.3	Example shortest phrases and their nearest neighbors across three layers.	84
7.1	Top activators of feature 4. Maximum activating words are shown in boldface.	103
7.2	Highest activating words of $(Wx)_4$ (top) and $(W^zx)_4$ (bottom). Ac- tivation amounts are given in parentheses.	106
7.3	Lowest activators of feature 9. Activations are coded with un- derline colors.	109
7.4	Highest activators of feature 2 after PCA. Activations are coded with underline colors.	115
7.5	Highest activators of feature 4 after PCA. Activations are coded with underline colors.	116
7.6	Top negative activators of feature 7 after PCA. Activations are coded with underline colors.	117
A.1	Top activators of feature 5. Activations are coded with underline colors.	125
A.2	Top activators of feature 14. Activations are coded with under- line colors.	127
A.3	Top negative activators of feature 5 after PCA. Activations are coded with underline colors.	127
A.4	Top activators of feature 7 after PCA. Activations are coded with underline colors.	128
A.5	Top activators of feature 10 after PCA. Activations are coded with underline colors.	128

LIST OF FIGURES

1.1	Distributed word representations	3
3.1	Recurrent neural networks. Each black, orange and red node denotes an input, hidden or output layer, respectively. Solid and dotted lines denote the connections of forward and backward layers, respectively. Top: Shallow unidirectional (left) and bidirectional (right) recurrent net. Bottom: 3-layer deep unidirectional (left) and bidirectional (right) recurrent net.	24
3.2	Examples of output. In each set, the gold-standard annotations are shown in the first line.	36
3.3	DEEP Recurrent Output vs. SHALLOW Recurrent Output. In each set of examples, the gold-standard annotations are shown in the first line. Tokens assigned a label of Inside with no preceding Begin tag are shown in ALL CAPS.	37
4.1	Vector x (blue) and tensor A (red) sliced along the dimension of x . Left. Dense word vector x computes a weighted sum over base matrices to get a square matrix, which then is used to transform the meaning vector. Right. One-hot word vector x with the same computation, which is equivalent to selecting one of the base matrices and falls back to a matrix-space model.	48
4.2	Hidden layer vectors reduced to 2 dimensions for various phrases. Left. Recurrent neural network. Right. Purely multiplicative recurrent neural tensor network. In mRNT, handling of negation is more nonlinear and correctly shifts the sentiment.	51
5.1	Structural recursive composition	56
5.2	Left. Recursive neural network with bottom-up propagation. Right. Bidirectional recursive neural network with bottom-up and top-down propagation. Black, orange and red denote bottom-up, top-down and output layers, respectively. Connections that share the same color and style are shared.	59
5.3	Experimental results for joint detection over sentences with separation	68
6.1	Operation of a recursive net (left), untied recursive net (middle) and a recurrent net (right) on an example sentence. Black, orange and red dots represent input, hidden and output layers, respectively. Directed edges having the same color-style combination denote shared connections.	73

6.2	Operation of a 3-layer deep recursive neural network . Red and black points denote output and input vectors, respectively; other colors denote intermediate memory representations. Connections denoted by the same color-style combination are shared (i.e. share the same set of weights).	75
6.3	An example sentence with its parse tree (left) and the response measure of every layer (right) in a three-layered deep recursive net. We change the word “ <i>best</i> ” in the input to one of the words “ <i>coolest</i> ”, “ <i>good</i> ”, “ <i>average</i> ”, “ <i>bad</i> ”, “ <i>worst</i> ” (denoted by blue, light blue, black, orange and red, respectively) and measure the change of hidden layer representations in one-norm for every node in the path.	83
7.1	Feature activators in computer vision and in NLP. Left. Highest activating image patches of a hidden unit of a single layer perceptron autoencoder. We can easily see the common pattern being edge-like (actual values for the splitting hyperplane shown below). Right. Highest activating sentences of a hidden unit of a recurrent net.	92
7.2	Some high activators of feature 4.	104
7.3	More high activators of feature 4.	105
7.4	Top two activators of feature 9.	107
7.5	Some low activators of feature 7.	108
7.6	Some instances that have high output deviation with the absence of feature 7.	110
7.7	Some aggregate statistics of 25 features. Top. Mean average immediate squared change (spikyness). Middle. Mean average update gate activation. Bottom. Mean average squared activation (squared distance to zero).	111
7.8	Some instances that have high activation of feature 17. Saliency and expected deviations differ.	112
7.9	Some instances that have high activation of feature 18. Saliency and expected deviations differ.	113
A.1	Some activations visualized with cross-feature saliency values (second block of columns). Most values are close to zero.	123
A.2	More activations visualized with cross-feature saliency values (second block of columns). Values are away from zero.	124
A.3	Activators of decorrelated feature 0 (top) and 1 (bottom)	126

NOTATION

$ x $	Dimensionality of vector x
$ S $	Cardinality of set S
$[X, Y]$	Horizontal concatenation of blocks X and Y
$[X; Y]$	Vertical concatenation of blocks X and Y
$\mathbb{1}(\cdot)$	Indicator function
$I(\cdot)$	Identity function
I	Identity matrix
$\mathbb{R}, \mathbb{R}^n, \mathbb{R}^{m \times n}$	Set of real numbers, n dimensional real vectors, and m by n real valued matrices, respectively
$A^{[i]}$	i th slice of a third order tensor A of $d_1 \times d_2 \times d_3$ (which is a $d_2 \times d_3$ matrix)
x, h, b, c	Scalars or (column) vectors
W, V, U, A	Matrices or higher order tensors
$\mathcal{X}, \mathcal{H}, \mathcal{V}, \mathcal{U}, \mathcal{S}$	Vector spaces, sets or probability distributions
$f(\cdot, \cdot)$	Composition function
$\sigma(\cdot)$	Nonlinearity of hidden layers in a neural network
$\gamma(\cdot)$	Nonlinearity of output layers in a neural network
t	Index of an element in a sequence
η	Index of a node in a tree

CHAPTER 1

INTRODUCTION

1.1 Overview

Recent advances in neural networks and deep learning have provided fruitful applications for natural language processing (NLP) tasks. Neural networks employ representation and feature learning, as opposed to feature engineering which is traditionally used in NLP (Manning et al., 2008). Such representation learning approaches reached new state-of-the-art on many problems in NLP, such as language modeling (Mikolov et al., 2011), sentiment analysis (Socher et al., 2013), machine translation (Cho et al., 2014) and others. Furthermore, these approaches remove the necessity of manual feature engineering which is often laborious and time consuming.

One especially important advance was the invention of word vector representations (Bengio et al., 2001). Word vectors represent every word in a dense, low dimensional space (see Figure 1.1), as opposed to traditional one-hot representation approach in which a word is assigned a vector of all zero values except a single one pointing to its index, which yields a sparse, high dimensional space. This representation also allows quantifying the similarity of different words, using the notion of distance. In turn, word vector representations have been employed in many applications of deep learning in NLP, as the main atomic unit to represent text (Collobert and Weston, 2008; Collobert et al., 2011).

Having strong tools for word representations, the natural next question, then, was how to properly map larger phrases into such dense representations

for NLP tasks that require properly capturing their meaning. Most existing methods take a compositional approach by defining a function that composes multiple word vector representations into representations of larger units of text, such as phrases or sentences (Mikolov et al., 2013b; Socher et al., 2013; Yessenalina and Cardie, 2011). These methods include orderless composition (similar to how bag-of-words combines one-hot word representations) (Mitchell and Lapata, 2010; Zanzotto et al., 2010; Mikolov et al., 2013b; Iyyer et al., 2015), sequential composition which combines words one-by-one in order (Rudolph and Giesbrecht, 2010; Yessenalina and Cardie, 2011; Mikolov et al., 2011) and structural composition, which combines words into larger and larger phrases in a bottom-up fashion, using dependency or constituency parse trees (Socher et al., 2011b, 2012b, 2013), among others.

My research goal in this dissertation is to investigate and improve such compositional approaches, particularly sequential and structural approaches to compositionality.

1.1.1 Distributed Word Representations

In this dissertation, word vectors constitute the main building blocks of language. Word vectors represent a word as a point in a dense, continuous, low-dimensional (typically on the order of 100s) vector space. This is in contrast to the traditional approach of representing a word as a one-hot vector, which yields a sparse and very high dimensional (as many as the total number of words in the dictionary) representation. Additionally, dense word vectors provide a means to measure similarity between different words using the notion of distance (or

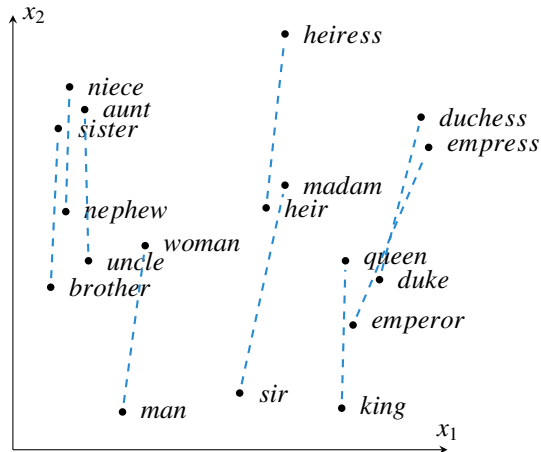


Figure 1.1: Distributed word representations

angle) whereas in one-hot word vectors every word has the same distance to every other word.

Word vectors are typically learned in an unsupervised fashion, which means we can employ large, unlabeled data sets for learning. Supervised approaches exist as well but they often do not have the benefit of having arbitrarily large data sets. Thus, in general, it is preferred to learn word vectors without supervision and then fine-tune them for the supervised task at hand.

Word vectors have been shown to be good semantic representation of words, on many tasks (Mikolov et al., 2013b; Socher et al., 2011b; Collobert et al., 2011; Iyyer et al., 2015). Additionally, word vector spaces have been shown to have interesting geometric properties connected to their relationships (Mikolov et al., 2013a,b; Levy and Goldberg, 2014a). For instance, in Figure 1.1, transition from male to female nouns occur roughly along the same vector of translation. These results and attractive properties of word vectors have let them to achieve widespread use in the NLP community (Socher et al., 2012a).

1.1.2 Composition of Word Vectors

Having a mechanism for representing words, the next stage is to represent phrases or sentences, which is the main focus of this thesis.

Compositional approaches to semantic interpretation state that the meaning of a phrase, or sentence is a composition of the meaning of its parts. This, in the context of word vectors, could be interpreted as a unit of text being represented by some function of its word vectors. This naturally brings us to the question of what compositional function to use to combine word vectors to represent the meaning of phrases or sentences.

A simple way to combine word vectors in a phrase, or sentence is to just use a commutative (orderless) operation, such as elementwise multiplication or averaging. This alone has been shown to be effective in certain applications such as sentiment analysis even though the order information is lost (Mikolov et al., 2013b; Iyyer et al., 2015).

Alternatively, we might define a composition function that is respectful of the order. For instance, we might define a recursive function that starts from the first word vector and combines it with the second word vector, then combines the result with the third word vector, and so on until the last.

Such a recursive approach could also be applied over structural information, in particular, parse trees of sentences. We can apply the function recursively, in a bottom up fashion, to get vector representations of phrases and sentences from the representations of their children.

In this thesis, these recursive functions that act as the main building block

for composition are modeled by neural networks.

1.1.3 Artificial Neural Networks

The main building block of a neural network is the perceptron: the application of a (typically sigmoidal) nonlinearity function on an affine transformation of the input. Deep neural networks involve cascading such perceptron layers, which allow multi-stage, hierarchical information processing through intermediate representations of the input.

Some of the advantages of neural networks over alternative learners are:

1. **Simplicity.** Many different architectures can be defined using the same building block and same formulation, all of which can be learned with the same algorithmic principle (backpropagation).
2. **Effectiveness.** Recent advances in computing power have resulted in the ability to train powerful neural networks that reached new state-of-the-art performance on many tasks ranging from computer vision to speech recognition.
3. **Feature learning.** Hierarchical representation learning in neural networks has shown the ability to combine simpler features into more and more complex and abstract features. In many cases, neural networks have eliminated the need to do manual feature engineering (Krizhevsky et al., 2012; Le et al., 2012).

Some drawbacks and challenges for neural networks involve:

1. Black-box nature. Although neural networks may have good performance, they are essentially black-box learners and usually difficult to interpret. This inability to manually inspect the function that is realized by the network makes error analysis difficult. Thus, in applications where manual inspection of the prediction by an expert is necessary, such as medical diagnosis, they become unsuitable. Lack of interpretability holds in the case of NLP as well, relative to fields such as computer vision, where features have visual meaning and are easy to visually inspect.
2. Data size requirements. Purely supervised training of neural networks usually requires large data sets, typically at least around the order of 1000s-10000s of training examples. That makes neural networks difficult to apply to tasks where annotated data, i.e. text paired with supervisory labels, is not so abundant, which is not uncommon in NLP.
3. Soft computing. The continuous representations employed by neural networks make them difficult to couple with symbolic, discrete approaches. Many problems in NLP are naturally formulated as discrete optimization problems which make neural networks difficult to apply, if not totally unsuitable.

This dissertation builds on top of existing neural network approaches that rely on sequential or structural interpretations of language. We investigate the effectiveness of such approaches for specific tasks of natural language understanding, and extend some of the approaches to provide representation hierarchies. We aim to successfully represent meaning both in isolation, and in context, depending on the requirements of the application. In part, we unify matrix-space models, which is another distributed compositional model without word vectors, with the recurrent neural network approach.

1.2 Contributions

The primary contribution of this dissertation is the development and investigation of effective models for representation learning for compositional meaning in natural language. More specifically, we make the following contributions:

Opinion mining without features using deep sequential models. We employ recurrent neural neural networks — a class of neural network with sequence modeling capabilities — to the task of opinion mining. In particular, we investigate the performance of shallow and deep (stacked) variants of the model. Our findings show that deep variants of the model consistently outperform the shallow variant. Furthermore, deep recurrent nets reach new state-of-the-art performance on the task, with no feature engineering, outperforming previous models that employ additional features from lexicons and parse trees. This work is described in Chapter 3.

Composition with multiplicative sequential models. We employ multiplicative recurrent neural networks, a variant of recurrent neural networks with additional multiplicative interactions, on the task of sentiment detection of phrases and sentences. For instance, one might intuit that negation can be modeled as multiplication by -1 (or, say, -0.5), such interactions can be captured more easily by explicitly allowing multiplicative interactions. In addition to yielding good performance on the task, we show that multiplicative models can be seen as a generalization of the matrix-space models. This work is presented in Chapter 4.

Learning structure-based representations of phrases in context. We propose a novel architecture to induce top-down representations for phrases and

words using parse trees of sentences. We apply the model on the task of opinion mining, posed as a sequence tagging task. We show that additional information brought by the structure improves the performance. This work is the subject of Chapter 5.

Deeper models for structural compositionality of sentences. Inspired by various kinds of deep neural networks of other types, we propose a deep variant of the recursive neural network. Deep recursive networks employ multi-stage information processing when composing smaller phrases into larger ones using the parse tree. The model achieves state-of-the-art performance on the task of sentence level sentiment detection. This work is given in Chapter 6.

Towards understanding neural models of composition. In order to make sense of the behavior of a compositional neural model trained over natural language, we investigate the features it learns to represent in its hidden layer. More concretely, similar to the analyses done in previous work in computer vision, we inspect instances that activate a given hidden unit the most. We identify challenges involved in this process which are specific to the context of natural language processing. We propose statistics that allow us to assign impact to individual words inside a sentence and visualize the activator sentences to reason about what feature a given hidden unit represents. This work is given in Chapter 7.

1.3 Roadmap

The rest of the dissertation is organized as follows. In Chapter 2 we give an overview of the background in neural networks and applications to composi-

tionality. The following two chapters involve sequential composition methods: In Chapter 3, we present an application of deep recurrent neural networks to the task of opinion mining. In Chapter 4, we apply multiplicative recurrent networks to sentiment analysis. Next two chapters discuss structural composition methods: In Chapter 5, we propose bidirectional recursive neural networks that can be applied to token-level tagging tasks in the structural setting. In Chapter 6, we propose deep recursive neural networks to incorporate feature hierarchies in structural neural composition. Finally, in Chapter 7 we identify challenges in understanding and interpreting compositional neural models in the context of natural language processing and propose simple first steps towards that goal.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we present an overview of the background and existing research in representation learning for natural language processing, as well as existing techniques for compositional approaches that are related to the work presented in this dissertation. We further discuss the related work in the context of specific compositional models and their applications in the corresponding chapters.

2.1 Representation Learning

In machine learning, first step is to decide on a representation of the data, which can greatly impact the overall success of the learner. Thus, many machine learning pipelines involve steps to preprocess the data of interest into useful representations that make it easier to extract relevant information. In natural language processing, for instance, considerable effort is spent on designing features: *Individual words or bigrams? How much context would be necessary? Is capitalization important? Would it help if we constructed a knowledge base of entities and their relations?* And so on. As seen from the examples, such processes of feature engineering may require a deep understanding of the task at hand by domain experts. They may be task dependent and fail to generalize to other tasks. Furthermore, making these decisions can be expensive, laborious and time consuming.

Representation learning, in contrast, attempts to automatically learn useful representations for data to partially or completely eliminate the step of man-

ual feature engineering (Bengio et al., 2013). In recent years, with the advancements in algorithms and technologies (such as GPUs), machine learning has seen a surge of very successful applications of representation learners in speech recognition (Hinton et al., 2012a; Deng et al., 2013), computer vision (Krizhevsky et al., 2012; Le et al., 2012; Egmont-Petersen et al., 2002), and natural language processing (Collobert and Weston, 2008; Collobert et al., 2011; Socher et al., 2012a), reaching or even outperforming feature-engineering oriented approaches.

2.2 Learning Word Representations

In natural language processing, a common way of representing a single token as a vector is to use a “one-hot” vector per token, with a dimensionality of the vocabulary size, such that the corresponding entry of the vector is 1, and all others are 0 (Manning and Schütze, 1999). This is a natural extension from how categorical features are typically represented in machine learning. Such a label oriented representation treats every word as a separate label, completely agnostic to any semantic content or relationship between words, i.e. the distance (or angle) between any two different word vectors is the same.

Distributional representations aim to solve this shortcoming by using cooccurrence statistics from text data. *The distributional hypothesis* suggests that words that occur in similar contexts share similar meaning (Harris, 1954). Therefore, a word vector representation that is derived by the cooccurrence matrix can contain semantic relationship information between different words, by way of quantifying the similarity of their contexts, hence, meanings. The exact form of

the cooccurrence matrix (raw counts vs. binary, size of the finite context window, etc.) or the way word vectors derived from the matrix (singular value decomposition, topic modeling, clustering, etc.) might differ (Turian et al., 2010). See Erk (2012) for a survey of distributional vector space models.

Distributed representations embed words in a dense, real valued, low-dimensional vector space. They are also typically *distributional*, since they are induced from cooccurrence statistics. The advantage of having distributed representations is that the encoding capacity grows exponentially with the dimensionality (Turian et al., 2010). There have been many advances in recent years proposing various ways to learn such word vectors, improving the efficiency of the methods and the quality of the resulting representations (Levy et al., 2015). They are, in general, learned from unsupervised text data, which is easy to obtain, and then fine-tuned for the end task at hand. Some approaches include context window based deep or shallow neural networks (Collobert and Weston, 2008; Mikolov et al., 2013b), sequential neural language models (Bengio et al., 2001; Mikolov et al., 2011), or explicit transformations of the cooccurrence matrix (Lebret and Collobert, 2013). In fact, some of the neural network based methods have been shown to implicitly factorize the cooccurrence matrix (Levy and Goldberg, 2014b). See Levy et al. (2015) for a comparison of some of the recent word vector learners.

Depending on the exact training mechanism, some classes of word vectors have been shown to have linear substructures in the embedding space, resulting in interesting geometric properties (Mikolov et al., 2013b). A famous example is that the embedded representation of the word *queen* can be roughly recovered

from the representations of *king*, *man* and *woman*:

$$queen \approx king - man + woman$$

Levy and Goldberg (2014a) investigate these structures and their properties. Pennington et al. (2014) explicitly formulate the objective function to incorporate such substructures in the word vector space.

Distributed word vectors have been applied to many NLP tasks, achieving new state of the art in many instances and having a big impact in the recent NLP literature (Turian et al., 2010; Levy et al., 2015).

So far, all of our discussion has focused on representing a word as a vector, which is generally called a **vector space model**. However, other representations can use more than just vectors. For instance, a **matrix space model** represents every word as a linear transformation, i.e. a matrix (Rudolph and Giesbrecht, 2010). In this model, words can be interpreted as functions that transform the meaning state. Baroni and Zamparelli (2010) propose a mixed representation where adjectives are matrices and nouns are vectors, to model adjective-noun compositions. Socher et al. (2012b) propose using both approaches where a word is assigned to both a matrix and a vector, to handle both its *transformer* and *transformee* representation.

2.3 Learning Representations for Phrases and Sentences

It is possible to learn representations for larger units of text directly, e.g. Le and Mikolov (2014b). However a more modular and intuitive approach is to start from word representations and use rules to combine them, employing linguistic

compositionality. The *principle of compositionality* asserts that the meaning of a complex expression is a function of the meanings of its constituent expressions and the rules used to combine them.

Some of the earlier work considered task-dependent mechanisms for handling specific modifiers. For instance, in sentiment analysis, negation (“not”) can simply invert the polarity of the sentiment decision (Choi and Cardie, 2008; Nakagawa et al., 2010), or dampen its intensity (Taboada et al., 2011; Liu and Seneff, 2009). Similarly, “very” can be considered as an intensifier. Such feature engineering approaches have been applied to bag-of-words representations to explicitly handle modifiers and update the sentiment of neighboring words (Polanyi and Zaenen, 2006; Kennedy and Inkpen, 2006; Shaikh et al., 2007). In contrast, compositional approaches attempt to define general universal rules to combine arbitrary parts, typically using distributional representations as their building blocks.

2.3.1 Orderless Composition

Perhaps the simplest way to compose words in an expression is to use an elementwise, commutative operation on word vectors. This way, resulting expression vector will lie in the same space as the word vectors themselves. For instance, Mitchell and Lapata (2008) investigate elementwise addition, or multiplication. Mikolov et al. (2013b) suggest vector averaging to represent short phrases. Even though such methods lose word order information, they can still be surprisingly powerful; Iyyer et al. (2015) show that feeding mean of all word vectors in the input sentence into deep feedforward neural networks can yield

very competitive results compared to complex state-of-the-art models in sentiment classification.

2.3.2 Sequential Composition

To incorporate word order information, one can define compositional rules that depend on the position of words with respect to one another. For instance, a composition function can start from the left, and recursively combine the intermediate representation so far:

$$h_t = f(h_{t-1}, x_t) \tag{2.1}$$

where h_t denotes the t th intermediate representation and x_t denotes the t th word representation. Because of the order in which we apply $f(\cdot, \cdot)$, if it is not symmetric with respect to its two argument, it will be a function of the word order.

Exact form of $f(\cdot, \cdot)$ may vary. Using neural networks to implement f provides attractive properties, such as end-to-end differentiability. A choice of a *perceptron* layer (or a single layer feedforward neural network) for definition of f results in the (*Elman-type*) *recurrent neural network* (Elman, 1990). Originally proposed for time-series prediction, recurrent networks can be applied to any spatio-temporal sequence in general, which can be a good representation of text when considered as a sequence of word vectors. Typically, they are trained using the *backpropagation through time* algorithm which is an extension of the standard *backpropagation* method to efficiently compute the gradients of a neural network (Werbos, 1990). See Bengio et al. (1994) for practical difficulties involving training.

By changing the exact form and definition of $f(\cdot, \cdot)$, we can design many different recurrent architectures. Among the many variants, Sutskever et al. (2011) propose *multiplicative* recurrent neural networks to incorporate multiplicative interactions between h and x . Hochreiter and Schmidhuber (1997) propose a gated variant that include *input*, *output* (and later *forget*) gates to explicitly control memory operations, named *long short-term memory* (LSTM). Cho et al. (2014) propose a simplified version named *gated recurrent unit* (GRU) that only has *update* and *reset* gates. Schmidhuber (1992), El Hihi and Bengio (1995) and Hermans and Schrauwen (2013) investigate *deep recurrent neural networks* by stacking multiple recurrent layers. See Jozefowicz et al. (2015) for an empirical comparison of some of these architectures.

In many application areas of machine learning that require spatio-temporal sequence processing, recurrent neural networks have found wide use in recent years. Recurrent neural networks have been applied to gesture recognition (Murakami and Taguchi, 1991), or stock price pattern recognition (Kamijo and Tanigawa, 1990). In speech recognition and acoustic modeling, LSTMs have reached state-of-the-art performance (Graves et al., 2013; Graves and Jaitly, 2014; Sak et al., 2015). In Gregor et al. (2015) LSTMs are used to sequentially attend to different parts of the canvas to generate images. *Multidimensional* recurrent neural networks have been applied to handwriting recognition, treating image pixels as a two-dimensional spatio-temporal sequence (Graves and Schmidhuber, 2009). In *Neural Turing Machines* an LSTM controller is used to make sequential decisions on external memory and learn arbitrary algorithms.

In the context of natural language processing, recurrent neural networks have reached wide usage as well, even though linguistic compositionality was

often not explicitly addressed. Many applications were on language modeling in which a recurrent network is trained to predict the next word in a sequence (Mikolov et al., 2010, 2011; Duh et al., 2013; Adel et al., 2013; Auli et al., 2013; Auli and Gao, 2014). Other applications include spoken language understanding (Mesnil et al., 2013), sequence tagging (Xu et al., 2015), sentiment analysis (Wang et al., 2015; Li et al., 2015c), dependency parsing (Dyer et al., 2015; Watanabe and Sumita, 2015), and text normalization (Chrupała, 2014).

In machine translation, sequence-to-sequence approaches have spanned a line of research where pairs of *encoder* and *decoder* recurrent networks are used. Sutskever et al. (2014) and Cho et al. (2014) use a recurrent encoder to compress an entire source sentence into a vector and then use a recurrent decoder to generate the target sentence from that vector. Bahdanau et al. (2014) extend this approach by providing an *attention mechanism* to the decoder network so that it can explicitly choose which parts of the encoder sequence representation to focus on. Vinyals et al. (2015) apply the same approach to constituency parsing, by representing the parse tree as a sequence and treating this sequence representation as target language.

Character level applications of recurrent networks have been of interest as well, even though in nature they might be different than the word level composition that is of interest in this dissertation. Many of the applications use character level language modeling or text generation as a test bed to evaluate the quality of the sequence modeler itself, without having an end-task to apply (Sutskever et al., 2011; Hermans and Schrauwen, 2013; Karpathy et al., 2015). Ling et al. (2015) apply character level composition to generate word representations in the open vocabulary setting, which can be especially useful for

morphologically rich languages such as Turkish.

Finally, an alternative sequential composition method to recurrent neural networks is the matrix-space model. Matrix-space models treat every word as a square matrix (Baroni and Zamparelli, 2010). Then, the representation of a phrase or sentence is given by the matrix multiplication of individual word matrices in order. Thus, semantic composition becomes function composition. Since matrix multiplication is not commutative, this operation preserves word order information. Note that this still fits Equation 2.1, when h and x are matrices and f is matrix multiplication. Yessenalina and Cardie (2011) have applied this model to ordinal sentiment classification.

2.3.3 Structural Composition

It is possible to change the order in which we compose things. For example, in the case where each *whole* has two parts, *left* and *right*, which can be viewed as a positional binary tree, we can make use of this structure to guide our composition function:

$$h_{\eta} = f(h_{left(\eta)}, h_{right(\eta)}) \quad (2.2)$$

Observe that Equation 2.1 is a special case of this in which *left* part is the prefix and *right* part is the next word that follows. Using a perceptron layer to model $f(\cdot, \cdot)$ results in the *recursive neural network* (Pollack, 1990). Recursive networks can be trained using *backpropagation through structure* to compute the gradients efficiently (Goller and Kuchler, 1996).

Similar to the recursive networks, modifying the exact implementation of f resulted in many different variants for the recursive neural networks. Socher

et al. (2012b) propose embedding words in a *matrix-vector space* and use matrix-vector multiplications in the definition to explicitly model how one word transforms the meaning of its sibling during composition. Socher et al. (2013) explore multiplicative variants by incorporating a tensor in the equation. Tai et al. (2015) propose a gated variant which is similar to the LSTM, that operates on tree structures.

Structure is abundant in natural language, in the form of constituency, dependency or discourse parse trees, therefore many applications of recursive neural networks used such representations. Socher et al. (2011b) used recursive networks to parse natural images and natural language sentences. Variations of recursive networks have been applied to dependency parsing (Le and Zuidema, 2014; Zhu et al., 2015a) and discourse parsing (Li et al., 2014). Socher et al. (2011a) applied them for paraphrase detection by way of measuring similarities between pairs of text. Luong et al. (2013) applied recursive networks to morpheme trees to exploit morphology information when producing word vector representations from morpheme vectors. Iyyer et al. (2014a) used dependency tree representations for question answering.

Perhaps the most popular applications were on sentiment classification (Socher et al., 2011c, 2013; Li et al., 2015c; Hermann and Blunsom, 2013). The proposal of the *Stanford Sentiment Treebank* dataset even further popularized this approach, because it contains sentiment scores for every phrase (sub-tree) in addition to sentence level scores, providing a test bed for recursive models (Socher et al., 2013). Li et al. (2015c) compare recursive and recurrent models to evaluate the effectiveness of having structure information.

CHAPTER 3

OPINION MINING WITH DEEP RECURRENT NEURAL NETWORKS

In this chapter, we present an application of deep recurrent neural networks to extract opinion expressions without relying on any feature engineering methods. The work described in this chapter is based on Irsoy and Cardie (2014b). We will first present a description of the task of opinion mining, then formulate the exact architecture definitions that we employ.

Fine-grained opinion analysis aims to detect the subjective expressions in a text (e.g. “hate”) and to characterize their intensity (e.g. strong) and sentiment (e.g. negative) as well as to identify the opinion holder (the entity expressing the opinion) and the target, or topic, of the opinion (i.e. what the opinion is about) (Wiebe et al., 2005). Fine-grained opinion analysis is important for a variety of NLP tasks including opinion-oriented question answering and opinion summarization. As a result, it has been studied extensively in recent years.

In this chapter, we focus on the detection of opinion expressions — both *direct subjective expressions* (DSEs) and *expressive subjective expressions* (ESEs) as defined in Wiebe et al. (2005). DSEs consist of explicit mentions of private states or speech events expressing private states; and ESEs consist of expressions that indicate sentiment, emotion, etc., without explicitly conveying them. An example sentence is shown in Table 3.1 in which the DSE “has refused to make any statements” explicitly expresses an opinion holder’s attitude and the ESE “as usual” indirectly expresses the attitude of the writer.

Opinion extraction has often been tackled as a sequence labeling problem in previous work (e.g. Choi et al. (2005)). Similar to our discussion in Chapter 1

3.1 Related Work

Opinion extraction. Early work on fine-grained opinion extraction focused on recognizing subjective phrases (Wilson et al., 2005; Munson et al., 2005). Breck et al. (2007), for example, formulated the problem as a token-level sequence-labeling problem and apply a CRF-based approach, which significantly outperformed previous baselines. Choi et al. (2005) extended the sequential prediction approach to jointly identify opinion holders; Choi and Cardie (2010) jointly detected polarity and intensity along with the opinion expression. Reranking approaches have also been explored to improve the performance of a single sequence labeler (Johansson and Moschitti, 2010, 2011). More recent work relaxes the Markovian assumption of CRFs to capture phrase-level interactions, significantly improving upon the token-level labeling approach Yang and Cardie (2012). In particular, Yang and Cardie (2013) propose a joint inference model to jointly detect opinion expressions, opinion holders and targets, as well as the relations among them, outperforming previous pipelined approaches.

Deep learning. Recurrent neural networks (Elman, 1990) constitute one important class of naturally deep architecture that has been applied to many sequential prediction tasks. In the context of NLP, recurrent neural networks view a sentence as a sequence of tokens and have been successfully applied to tasks such as language modeling (Mikolov et al., 2011) and spoken language understanding (Mesnil et al., 2013). Since classical recurrent neural networks only incorporate information from the past (i.e. preceding tokens), *bidirectional* variants have been proposed to incorporate information from both the past and the future (i.e. subsequent tokens) (Schuster and Paliwal, 1997). Bidirectionality is especially useful for NLP tasks, since information provided by the following to-

kens is generally helpful (and sometimes essential) when making a decision on the current token.

Stacked recurrent neural networks have been proposed as a way of constructing *deep* recurrent networks (Schmidhuber, 1992; El Hihi and Bengio, 1995). Careful empirical investigation of this architecture showed that multiple layers in the stack can operate at different time scales (Hermans and Schrauwen, 2013). Pascanu et al. (2013) explore other ways of constructing deep recurrent networks that are orthogonal to the concept of stacking layers on top of each other. In this chapter, we focus on the *stacking* notion of depth.

3.2 Sequential Neural Models for Opinion Mining

This section describes the sequential compositional architectures and training methods that we apply to the task of opinion expression mining. Recurrent neural networks are presented in 3.2.1, bidirectionality is introduced in 3.2.2, and deep bidirectional recurrent nets, in 3.2.3.

3.2.1 Recurrent Neural Networks

A recurrent neural network (Elman, 1990) is a class of neural network that has recurrent connections, which allow a form of memory. This makes them applicable for sequential prediction tasks with arbitrary spatio-temporal dimensions. Thus, their structure fits many NLP tasks, when the interpretation of a single sentence is viewed as analyzing a sequence of tokens. In this work, we focus our attention on only Elman-type networks (Elman, 1990).

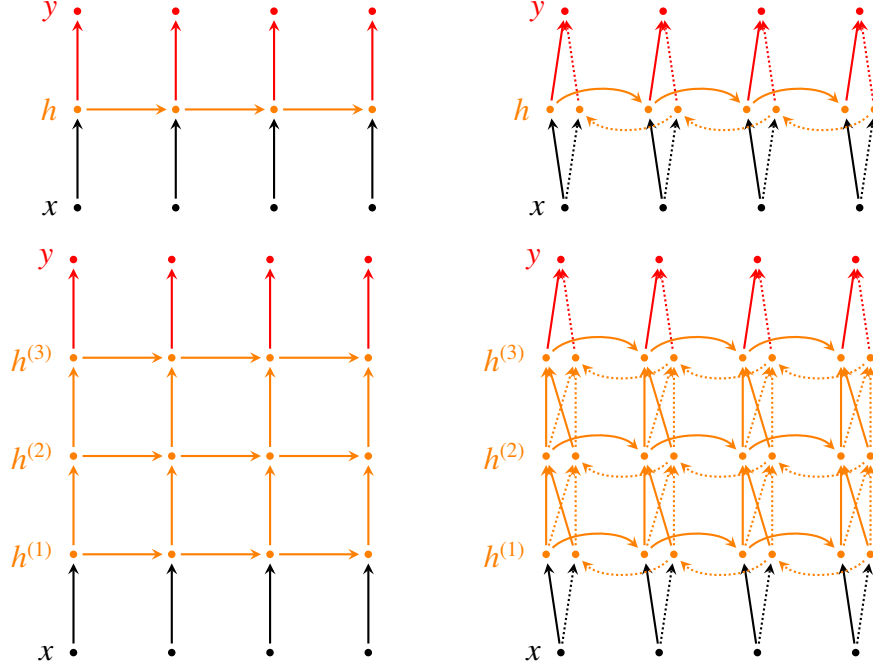


Figure 3.1: Recurrent neural networks. Each black, orange and red node denotes an input, hidden or output layer, respectively. Solid and dotted lines denote the connections of forward and backward layers, respectively. Top: Shallow unidirectional (left) and bidirectional (right) recurrent net. Bottom: 3-layer deep unidirectional (left) and bidirectional (right) recurrent net.

In an *Elman-type recurrent neural network*, the hidden layer h_t at time step t is computed from a nonlinear transformation of the current input layer x_t and the previous hidden layer h_{t-1} . Then, the final output y_t is computed using the hidden layer h_t . One can interpret h_t as an intermediate representation summarizing the past, which is used to make a final decision on the current input.

More formally, given a sequence of vectors $\{x_t\}_{t=1}^T$, an Elman-type recurrent network operates by computing the following memory and output sequences:

$$h_t = \sigma(Wx_t + Vh_{t-1} + b) \quad (3.1)$$

$$y_t = \gamma(Uh_t + c) \quad (3.2)$$

where σ is a nonlinear function, such as the sigmoid function and γ is the out-

put nonlinearity, such as the softmax function. W and V are weight matrices between the input and hidden layer, and among the hidden units themselves (connecting the previous intermediate representation to the current one), respectively, while U is the output weight matrix. b and c are bias vectors connected to hidden and output units, respectively. As a base case for the recursion in Equation 3.1, h_0 is assumed to be 0 (alternatively, h_0 can be left as a parameter of the model to be learned).

Training a recurrent network can be done by optimizing a discriminative objective (e.g. the cross entropy for classification tasks) with a gradient-based method. *Backpropagation through time* can be used to efficiently compute the gradients (Werbos, 1990). This method is essentially equivalent to unfolding the network in time and using backpropagation as in feedforward neural networks, while sharing the connection weights across different time steps.

The Elman-style recurrent network is shown in Figure 3.1, top left.

3.2.2 Bidirectionality

Observe that with the above definition of recurrent networks, we have information only about the past, when making a decision on x_t . This is limiting for most NLP tasks. As a simple example, consider the two sentences: “*I did not accept his suggestion*” and “*I did not go to the rodeo*”. The first has a DSE phrase (“*did not accept*”) and the second does not. However, any such network will assign the same labels for the words “did” and “not” in both sentences, since the preceding sequences (past) are the same: the Elman-style unidirectional recurrent networks lack the representational power to model this task. A simple way

to work around this problem is to include a fixed-size future context around a single input vector (token). However, this approach requires tuning the context size, and ignores future information from outside of the context window. Another way to incorporate information about the future is to add bidirectionality to the architecture, referred as the *bidirectional recurrent neural network* (Schuster and Paliwal, 1997):

$$\vec{h}_t = \sigma(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b}) \quad (3.3)$$

$$\overleftarrow{h}_t = \sigma(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b}) \quad (3.4)$$

$$y_t = \gamma(U_{\rightarrow}\vec{h}_t + U_{\leftarrow}\overleftarrow{h}_t + c) \quad (3.5)$$

where \vec{W} , \vec{V} and \vec{b} are the forward weight matrices and bias vector as before; \overleftarrow{W} , \overleftarrow{V} and \overleftarrow{b} are their backward counterparts; U_{\rightarrow} , U_{\leftarrow} are the output matrices; and c is the output bias. Again, we assume $\vec{h}_0 = \overleftarrow{h}_{T+1} = 0$. In this setting \vec{h}_t and \overleftarrow{h}_t can be interpreted as a summary of the past, and the future, respectively, around the time step t . When we make a decision on an input vector, we employ the two intermediate representations \vec{h}_t and \overleftarrow{h}_t of the past and the future. (See Figure 3.1, top right.) Therefore in the bidirectional case, we have perfect information about the sequence (ignoring the practical difficulties about capturing long term dependencies, caused by vanishing gradients), whereas the classical Elman-type network uses only partial information as described above.

Note that the forward and backward parts of the network are independent of each other until the output layer when they are combined. This means that during training, after backpropagating the error terms from the output layer to the forward and backward hidden layers, the two parts can be thought of as separate, and each trained with the classical backpropagation through time Werbos (1990).

3.2.3 Depth in Space

Recurrent neural networks are often characterized as having *depth in time*: when unfolded, they are equivalent to feedforward neural networks with as many hidden layers as the number tokens in the input sequence (with shared connections across multiple layers of time). However, this notion of depth likely does not involve hierarchical processing of the data: across different time steps, we repeatedly apply the same transformation to compute the memory contribution of the input (W), to compute the response value from the current memory (U) and to compute the next memory vector from the previous one (V). Therefore, assuming the input vectors $\{x_t\}$ together lie in the same representation space, as do the output vectors $\{y_t\}$, hidden representations $\{h_t\}$ lie in the same space as well. As a result, they do not necessarily become more and more abstract, hierarchical representations of one another as we traverse in time. However in the more conventional, *stacked* deep learners (e.g. deep feedforward nets), an important benefit of depth is the hierarchy among hidden representations: every hidden layer conceptually lies in a different representation space, and constitutes a more abstract and higher-level representation of the input (Bengio, 2009).

In order to address these concerns, we investigate deep recurrent neural networks, which are constructed by stacking Elman-type recurrent networks on top of each other (Hermans and Schrauwen, 2013). Intuitively, every layer of the deep recurrent network treats the memory sequence of the previous layer as the input sequence, and computes its own memory representation.

More formally, we have:

$$\vec{h}_t^{(i)} = \sigma(\vec{W}_{\rightarrow}^{(i)} \vec{h}_t^{(i-1)} + \vec{W}_{\leftarrow}^{(i)} \overleftarrow{h}_t^{(i-1)} + \vec{V}^{(i)} \vec{h}_{t-1}^{(i)} + \vec{b}^{(i)}) \quad (3.6)$$

$$\overleftarrow{h}_t^{(i)} = \sigma(\overleftarrow{W}_{\rightarrow}^{(i)} \vec{h}_t^{(i-1)} + \overleftarrow{W}_{\leftarrow}^{(i)} \overleftarrow{h}_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)}) \quad (3.7)$$

when $i > 1$ and

$$\vec{h}_t^{(1)} = \sigma(\vec{W}^{(1)} x_t + \vec{V}^{(1)} \vec{h}_{t-1}^{(1)} + \vec{b}^{(1)}) \quad (3.8)$$

$$\overleftarrow{h}_t^{(1)} = \sigma(\overleftarrow{W}^{(1)} x_t + \overleftarrow{V}^{(1)} \overleftarrow{h}_{t+1}^{(1)} + \overleftarrow{b}^{(1)}) \quad (3.9)$$

Importantly, note that both forward and backward representations are employed when computing the forward and backward memory of the next layer.

Two alternatives for the output layer computations are to employ all memory layers or only the last. In this work we adopt the second approach:

$$y_t = \gamma(U_{\rightarrow} \vec{h}_t^{(\ell)} + U_{\leftarrow} \overleftarrow{h}_t^{(\ell)} + c) \quad (3.10)$$

where ℓ is the number of layers. Intuitively, connecting the output layer to only the last hidden layer forces the architecture to capture enough high-level information at the final layer for producing the appropriate output-layer decision.

Training a deep recurrent network can be conceptualized as interleaved applications of the conventional backpropagation across multiple layers, and backpropagation through time within a single layer.

The unidirectional and bidirectional deep recurrent networks are depicted in the bottom half of Figure 3.1.

3.3 Experiments

3.3.1 Hypotheses

In general, we expected that the deep recurrent networks would show the most improvement over shallow recurrent networks for ESEs — phrases that implicitly convey subjectivity. Existing research has shown that these are harder to identify than direct expressions of subjectivity (DSEs): they are variable in length and involve terms that, in many (or most) contexts, are neutral with respect to sentiment and subjectivity. As a result, models that do a better job interpreting the context should be better at disambiguating subjective vs. non-subjective uses of phrases involving common words (e.g. “as usual”, “in fact”).

Whether or not deep recurrent networks would be powerful enough to outperform the state-of-the-art semiCRF was unclear, especially if the semiCRF is given access to the distributed word representations (embeddings) employed by the deep recurrent networks. In addition, the semiCRF has access to parse tree information and opinion lexicons, neither of which is available to the deep recurrent networks.

3.3.2 Experimental Setting

Activation Units. We employ the standard softmax activation for the output layer: $\gamma(x) = e^{x_i} / \sum_j e^{x_j}$. For the hidden layers we use the rectifier linear activation: $\sigma(x) = \max\{0, x\}$. Experimentally, rectifier activation gives better performance, faster convergence, and sparse representations. Previous work also re-

ported good results when training deep neural networks using rectifiers, without a pretraining step (Glorot et al., 2011).

Data. We use the MPQA 1.2 corpus (Wiebe et al., 2005) (535 news articles, 11,111 sentences) that is manually annotated with both DSEs and ESEs at the phrase level. As in previous work, we separate 135 documents as a development set and employ 10-fold CV over the remaining 400 documents. The development set is used during cross validation to do model selection.

Evaluation Metrics. We use precision, recall and F-measure for performance evaluation. Since the boundaries of expressions are hard to define even for human annotators (Wiebe et al., 2005), we use two soft notions of the measures: *Binary Overlap* counts every overlapping match between a predicted and true expression as correct (Breck et al., 2007; Yang and Cardie, 2012), and *Proportional Overlap* imparts a partial correctness, proportional to the overlapping amount, to each match (Johansson and Moschitti, 2010; Yang and Cardie, 2012). All statistical comparisons are done using a two-sided paired t-test with a confidence level of $\alpha = .05$.

Baselines (CRF and SEMICRF). As baselines, we use the CRF-based method of Breck et al. (2007) and the SEMICRF-based method of Yang and Cardie (2012), which is the state-of-the-art in opinion expression extraction. Features that the baselines use are words, part-of-speech tags and membership in a manually constructed opinion lexicon (within a $[-1, +1]$ context window). Since SEMICRF relaxes the Markovian assumption and operates at the segment-level instead of the token-level, it also has access to parse trees of sentences to generate candidate segments (Yang and Cardie, 2012).

Word Vectors (+VEC). We also include versions of the baselines that have access to pre-trained word vectors. In particular, CRF+VEC employs word vectors as continuous features per every token. Since SEMICRF has phrase-level rather than word-level features, we simply take the mean of every word vector for a phrase-level vector representation for SEMICRF+VEC as suggested in Mikolov et al. (2013b).

In all of our experiments, we keep the word vectors fixed (i.e. do not fine-tune) to reduce the degree of freedom of our models. We use the publicly available 300-dimensional word vectors of Mikolov et al. (2013b), trained on part of the Google News dataset (~ 100 B words). Preliminary experiments with other word vector representations such as Collobert-Weston (Collobert and Weston, 2008) embeddings or HLBL (Mnih and Hinton, 2007) provided poorer results ($\sim -3\%$ difference in proportional and binary F1).

Regularizer. We do not employ any regularization for smaller networks ($\sim 24,000$ parameters) because we have not observed strong overfitting (i.e. the difference between training and test performance is small). Larger networks are regularized with the recently proposed dropout technique (Hinton et al., 2012b): we randomly set entries of hidden representations to 0 with a probability called the dropout rate, which is tuned over the development set. Dropout prevents learned features from co-adapting, and it has been reported to yield good results when training deep neural networks (Krizhevsky et al., 2012; Dahl et al., 2013).

Network Training. We use the standard multiclass cross-entropy as the objective function when training the neural networks. We use stochastic gradient descent with momentum with a fixed learning rate (.005) and a fixed momen-

tum rate (.7). We update weights after minibatches of 80 sentences. We run 200 epochs for training. Weights are initialized from small random uniform noise. We experiment with networks of various sizes, however we have the same number of hidden units across multiple forward and backward hidden layers of a single recurrent network. We do not employ a pre-training step; deep architectures are trained with the supervised error signal, even though the output layer is connected to only the final hidden layer. With these configurations, every architecture successfully converges without any oscillatory behavior. Additionally, we employ early stopping for the neural networks: out of all iterations, the model with the best development set performance (Proportional F1) is selected as the final model to be evaluated.

3.3.3 Results and Discussion

Layers (ℓ)	$ h $	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop	Bin.
Shallow	36	62.24	65.90	65.63*	73.89*	63.83	69.62
Deep 2	29	63.85*	67.23*	65.70*	74.23*	64.70*	70.52*
Deep 3	25	63.53*	67.67*	65.95*	73.87*	64.57*	70.55*
Deep 4	22	64.19*	68.05*	66.01*	73.76*	64.96*	70.69*
Deep 5	21	60.65	61.67	56.83	69.01	58.60	65.06
Shallow	200	62.78	66.28	65.66*	74.00*	64.09	69.85
Deep 2	125	62.92*	66.71*	66.45*	74.70*	64.47	70.36
Deep 3	100	65.56*	69.12*	66.73*	74.69*	66.01*	71.72*
Deep 4	86	61.76	65.64	63.52	72.88*	62.56	69.01
Deep 5	77	61.64	64.90	62.37	72.10	61.93	68.25

Table 3.2: Experimental evaluation of recurrent networks for DSE extraction

Bidirectional vs. Unidirectional. Although our focus is on bidirectional recurrent networks, we first confirm that the SHALLOW bidirectional network out-

Layers (ℓ)	$ h $	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop	Bin.
Shallow	36	51.34	59.54	57.60	72.89*	54.22	65.44
Deep 2	29	51.13	59.94	61.20*	75.37*	55.63*	66.64*
Deep 3	25	53.14*	61.46*	58.01	72.50	55.40*	66.36*
Deep 4	22	51.48	60.59*	59.25*	73.22	54.94	66.15*
Deep 5	21	49.67	58.42	48.98	65.36	49.25	61.61
Shallow	200	52.20*	60.42*	58.11	72.64	54.75	65.75
Deep 2	125	51.75*	60.75*	60.69*	74.39*	55.77*	66.79*
Deep 3	100	52.04*	60.50*	61.71*	76.02*	56.26*	67.18*
Deep 4	86	50.62*	58.41*	53.55	69.99	51.98	63.60
Deep 5	77	49.90*	57.82	52.37	69.13	51.01	62.89

Table 3.3: Experimental evaluation of recurrent networks for ESE extraction

	Model	Precision		Recall		F1	
		Prop.	Bin.	Prop.	Bin.	Prop	Bin.
DSE	CRF	74.96*	82.28*	46.98	52.99	57.74	64.45
	semiCRF	61.67	69.41	67.22*	73.08*	64.27	71.15*
	CRF +vec	74.97*	82.43*	49.47	55.67	59.59	66.44
	semiCRF +vec	66.00	71.98	60.96	68.13	63.30	69.91
	Deep RNT 3 100	65.56	69.12	66.73*	74.69*	66.01*	71.72*
ESE	CRF	56.08	68.36	42.26	51.84	48.10	58.85
	semiCRF	45.64	69.06	58.05	64.15	50.95	66.37*
	CRF +vec	57.15*	69.84*	44.67	54.38	50.01	61.01
	semiCRF +vec	53.76	70.82*	52.72	61.59	53.10	65.73
	Deep RNT 3 100	52.04	60.50	61.71*	76.02*	56.26*	67.18*

Table 3.4: Comparison of Deep recurrent networks to state-of-the-art (semi)CRF baselines for DSE and ESE detection

performs a (shallow) unidirectional network for both DSE and ESE recognition. To make the comparison fair, each network has the same number of total parameters: we use 65 hidden units for the unidirectional, and 36 for the bidirectional network, respectively. Results are as expected: the bidirectional network obtains higher F1 scores than the unidirectional network — 63.83 vs. 60.35 (proportional overlap) and 69.62 vs. 68.31 (binary overlap) for DSEs; 54.22 vs. 51.51

(proportional) and 65.44 vs. 63.65 (binary) for ESEs. All differences are statistically significant at the 0.05 level. Thus, we will not include comparisons to the unidirectional nets in the remaining experiments.

Adding Depth. Next, we quantitatively investigate the effects of adding depth to recurrent networks. Tables 3.2 and 3.3 show the evaluation of recurrent networks of various depths and sizes. In both tables, the first group networks have approximately 24,000 parameters and the second group networks have approximately 200,000 parameters. Since all networks within a group have approximately the same number of parameters, they grow narrower as they get deeper. Within each group, bold shows the best result with an asterisk denoting statistically indistinguishable performance with respect to the best. As noted above, all statistical comparisons use a two-sided paired t-test with a confidence level of $\alpha = .05$.

In both DSE and ESE detection and for larger networks (bottom set of results), 3-layer networks provide the best results. For smaller networks (top set of results), 2, 3 and 4-layer networks show equally good performance for certain sizes and metrics and, in general, adding additional layers degrades performance. This could be related to how we train the architectures as well as to the decrease in width of the networks. In general, we observe a trend of increasing performance as we increase the number of layers, until a certain depth.

Deep recurrent net vs. (semi)CRF. Table 3.4 shows comparison of the best deep recurrent networks to the previous best results in the literature. In terms of F-measure, DEEP RNT performs best for both DSE and ESE detection, achieving a new state-of-the-art performance for the more strict proportional overlap measure, which is harder to improve upon than the binary evaluation metric.

SEMICRF, with its very high recall, performs comparably to the DEEP RNT on the binary metric. Note that recurrent networks do not have access to any features other than word vectors.

In general, CRFs exhibit high precision but low recall (CRFs have the best precision on both DSE and ESE detection) while SEMICRFs exhibit a high recall, low precision performance. Compared to SEMICRF, the DEEP RNTs produce an even higher recall but sometimes lower precision for ESE detection. This suggests that the methods are complementary, and can potentially be even more powerful when combined in an ensemble method.

Word vectors. Word vectors help CRFs on both precision and recall on both tasks. However, SEMICRFs become more conservative with word vectors, producing higher precision and lower recall on both tasks. This sometimes hurts overall F-measure.

Among the (SEMI)CRF-based methods, SEMICRF obtains the highest F1 score for DSEs and for ESEs using the softer metric; SEMICRF+VEC performs best for ESEs according to the stricter proportional overlap measure.

Network size. Finally, we observe that even small networks (such as 4-layer deep recurrent network for DSE and 2-layer deep recurrent network for ESE) outperform conventional CRFs. This suggests that with the help of good word vectors, we can train compact but powerful sequential neural models.

When examining the output, we see some systematic differences between the previously top-performing SEMICRF and the neural models. (See Figure 3.2.) First, SEMICRF often identifies excessively long subjective phrases as in Example 1. Here, none of the models exactly matches the gold standard, but

	(1)	The situation obviously remains fluid from hour to hour but it [seems to be] [going in the right direction]
DEEPRNT		The situation [obviously] remains fluid from hour to hour but it [seems to be going in the right] direction
SHALLOW		The situation [obviously] remains fluid from hour to hour but it [seems to be going in] the right direction
SEMICRF		The situation [obviously] remains fluid from hour to hour but it seems to be going in the right direction]
	(2)	have always said this is a multi-faceted campaign [but equally] we have also said any future military action [would have to be based on evidence] , ...
DEEPRNT		have always said this is a multi-faceted campaign but [equally we] have also said any future military action [would have to be based on evidence] , ...
SHALLOW		have always said this is a multi-faceted [campaign but equally we] have also said any future military action would have to be based on evidence , ...
SEMICRF		have always said this is a multi-faceted campaign but equally we have also said any future military action would have to be based on evidence , ...
	(3)	Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was [not yet] secure for aid agencies to operate in and “ [not enough] ” food had been taken into the country .
DEEPRNT		Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was [not yet] secure for aid agencies to operate in and “ [not enough] ” food had been taken into the country .
SHALLOW		Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was [not yet] secure for aid agencies to operate in and “ [not enough] ” food had been taken into the country .
SEMICRF		Ruud Lubbers , the United Nations Commissioner for Refugees , said Afghanistan was not yet secure for aid agencies to operate in and “ not enough ” food had been taken into the country .

Figure 3.2: Examples of output. In each set, the gold-standard annotations are shown in the first line.

the recurrent networks are much closer. And all three models appear to have identified an ESE that was mistakenly omitted by the human annotator — “obviously”. At the same time, the SEMICRF sometimes entirely misses subjective expressions that the recurrent nets identify — this seems to occur when there are no clear indications of sentiment in the subjective expression. The latter can be seen in Examples 2 and 3, in which the SEMICRF does not identify “but equally”, “would have to be based on evidence”, “not yet”, and “not enough”.

(4)	[In any case] , [it is high time] that a social debate be organized ...
DEEP	[In any case] , it is HIGH TIME that a social debate be organized ...
SHALLOW	In ANY case , it is high TIME that a social debate be organized ...
(5)	Mr. Stoiber [has come a long way] from his refusal to [sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances] run against the wishes...
DEEP	Mr. Stoiber [has come a long way from] his [refusal to sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances run against] the wishes...
SHALLOW	Mr. Stoiber has come A LONG WAY FROM his refusal to sacrifice himself for the CDU in an election that [once looked impossible] to win , through his statement that he would under NO CIRCUMSTANCES run against the wishes...

Figure 3.3: DEEP Recurrent Output vs. SHALLOW Recurrent Output. In each set of examples, the gold-standard annotations are shown in the first line. Tokens assigned a label of Inside with no preceding Begin tag are shown in ALL CAPS.

We also observe evidence of the power of the DEEP recurrent net over the SHALLOW recurrent net in Examples 4 and 5. (See Figure 3.3.) In contrast to Figure 3.2, Figure 3.3 distinguishes subjective expressions that are (correctly) assigned an initial Begin label from those that consist only of Inside labels¹ — the latter are shown in ALL CAPS and indicate some degree of confusion in the model that produced them. In Example 4, SHALLOW network exhibits *some* evidence for each ESE — it labels one or more tokens as Inside an ESE (“any” and “time”). But it does not explicitly tag the beginning of the ESE. DEEP network does better, identifying the first ESE in its entirety (“in any case”) and identifying more words as being Inside the second ESE (“it is high time”). A similar situation occurs in Example 5.

¹Sequences of I’s are decoded as the associated DSE or ESE even though they lack the initial B.

3.4 Chapter Summary

In this chapter we have explored an application of deep recurrent neural networks to the task of sentence-level opinion expression extraction. We empirically evaluated deep recurrent nets against conventional, shallow recurrent nets that have only a single hidden layer. We also compared our models with previous (semi)CRF-based approaches.

Experiments showed that deep networks outperformed shallow networks on both DSE and ESE extraction. Furthermore, deep recurrent networks outperformed previous (semi)CRF baselines, achieving new state-of-the-art results for fine-grained opinion expression extraction. This shows that a representation learning approach that is based on sequential composition via recurrent neural networks works as a better alternative to feature engineering approaches that are based on CRFs.

CHAPTER 4

MODELING COMPOSITIONALITY WITH MULTIPLICATIVE RECURRENT NEURAL NETWORKS

In this chapter we investigate multiplicative recurrent neural networks for the task of sentiment analysis and discuss how they are connected to an alternative sequential composition method: matrix-space models. This work was published in Irsoy and Cardie (2015).

As we have discussed in the previous chapter, *recurrent neural networks* (RNTs), a neural network architecture with sequential prediction capabilities, *implicitly* model compositionality when applied to natural language sentences. Representation of a phrase can be conceptualized as a nonlinear function that acts on the network’s hidden layer (memory), which results from repeated function composition over the hidden layer and the next word in the phrase/sentence (see Section 3.2.1). Unfortunately, it is possible that conventional additive recurrent networks are not powerful enough to accommodate some of the more complex effects in language, as suggested in previous work on (multiplicative and additive variants of) recursive neural networks (e.g. Socher et al. (2013)). More specifically, even though additive models can theoretically model arbitrary functions when combined with a nonlinearity, they might require a very large number of hidden units, since the main building blocks (hidden units) are simple linear hyperplane splits and the complexity of the function that can be modeled depends on the number of these building blocks. Thus, learnability of large parameter sets from data might pose an issue.

To this end we investigate the *multiplicative recurrent neural network* as a model for compositional semantic effects in language. Previously, this type of

multiplicative sequential approach has been applied to a character-level text generation task (Sutskever et al., 2011). In this work, we investigate its capacity for recognizing the sentiment of a sentence or a phrase represented as a sequence of dense word vectors. Like the matrix-space models, multiplicative RNTs are sequential models of language; and as a type of recurrent network, they implicitly model compositionality. Like the very successful multiplicative recursive neural networks (Socher et al., 2013), multiplicative RNTs can capture the same types of sibling interactions, but are much simpler. In particular, no parse trees are required, so sequential computations replace the associated recursive computations and performance does not depend on the accuracy of the parser.

We also show a connection between the multiplicative RNT and compositional matrix-space models, which have also been applied to sentiment analysis (Rudolph and Giesbrecht, 2010; Yessenalina and Cardie, 2011). In particular, we show that matrix-space models are effectively a special case of multiplicative RNTs in which a word is represented as a large “one-hot” vector instead of a dense small one. Thus, these networks carry over the idea of matrix-space models from a one-hot sparse representation to dense word vectors. They can directly employ word vector representations, which makes them better suited for semi-supervised learning given the plethora of word vector training schemes. Multiplicative recurrent networks can be considered to unify these two views of distributed language processing — the operator semantics view of matrix-space models in which a word is interpreted as an operator acting on the meaning representation, and the sequential memory processing view of recurrent neural networks.

4.1 Related Work

Vector Space Models. As previously discussed, a distributed word vector representation maps a token to a real-valued dense vector of small dimensionality (usually on the order of 100 dimensions). Generally, these representations are learned in an unsupervised manner from a large corpus, e.g. Wikipedia. Various architectures have been explored to learn these embeddings (Bengio et al., 2001; Collobert and Weston, 2008; Mnih and Hinton, 2007; Mikolov et al., 2013a) which might have different generalization capabilities depending on the task (Turian et al., 2010). The geometry of the induced word vector space might have interesting semantic properties ($king - man + woman \approx queen$) (Mikolov et al., 2013a,b).

Matrix Space Models. An alternative approach is to embed words into a matrix space, by assigning matrices to words. Intuitively, a matrix embedding of a word is desired in order to capture operator semantics: the embedding should model how a word transforms meaning when it is applied to a context. Baroni and Zamparelli (2010) partially apply this idea to model adjectives as matrices that act on noun vectors. In their theoretical work, Rudolph and Giesbrecht (2010) define a proper matrix space model by assigning every word to a matrix; representations for longer phrases are computed by matrix multiplication. They show that matrix space models generalize vector space models and argue that they are neurologically and psychologically plausible. Yessenalina and Cardie (2011) apply this model to fine-grained sentiment detection. Socher et al. (2012b) use a structural approach in which every word is assigned a matrix-vector pair, where the vector captures the meaning of the word in isolation and the matrix captures how it transforms meaning when applied to a vector.

Compositionality in Vector and Matrix Spaces. Commutative vector operations such as addition (e.g. bag-of-words) or element-wise multiplication along with negation (Widdows, 2003) provide simple composition schemes (Mitchell and Lapata, 2010; Zanzotto et al., 2010). Even though they ignore the order of the words, they might prove effective depending on the length of the phrases, and on the task (Mikolov et al., 2013b). Other models for compositional distributional semantics emulate formal semantics by representing functions as tensors and arguments as vectors (e.g. (Clark, 2008; Coecke et al., 2010; Grefenstette et al., 2013)) for which (Grefenstette et al., 2013) generalise the tensor-learning approach of (Baroni and Zamparelli, 2010). More complex non-commutative composition functions can be modeled via sequential or structural models of the sentence. In particular, compositionality in recurrent neural networks can be considered as transformations on the memory (hidden layer) applied by successive word vectors in order. In matrix space models, compositionality is naturally modeled via function composition in sequence (Rudolph and Giesbrecht, 2010; Yessenalina and Cardie, 2011).

Sentiment Analysis. Sentiment analysis has been a very active area among NLP researchers, at various granularities such as the word-, phrase-, sentence- or document-level (Pang and Lee, 2008). Besides preexisting work that tried to formulate the problem as binary classification, recently fine-grained approaches were explored (Yessenalina and Cardie, 2011; Socher et al., 2013). Ultimately, the vast majority of approaches do not tackle the task compositionally, and in addition to bag-of-words features, they incorporate engineered features to account for negators, intensifiers and contextual valence shifters (Polanyi and Zaenen, 2006; Wilson et al., 2005; Kennedy and Inkpen, 2006; Shaikh et al., 2007).

4.2 Composition with Matrix-Space Models

A matrix-space model models a single word as a square matrix that transforms a meaning (state) vector to another vector in the same meaning space. Intuitively, a word is viewed as a function, or an operator (in this particular case, linear) that acts on the meaning representation. Therefore, a phrase (or any sequence of words) is represented as successive application of the individual operators inside the phrase.

Let $s = w_1, w_2, \dots, w_T$ be a sequence of words of length T and let $M_w \in \mathbb{R}^{m \times m}$ denote the matrix representation of a word $w \in \mathcal{V}$ where \mathcal{V} is the vocabulary. Then, the representation of s is simply

$$M(s) = M_{w_1} M_{w_2} \dots M_{w_T} \quad (4.1)$$

which yields another linear transformation in the same space. Observe that this representation respects word order (unlike, e.g. a bag of words). Note that even though $M(s)$ is modeled as a linear operator on the meaning space, $M(s)$ as a function of $\{M_{w_i}\}_{i=1}^T$ is not linear, since it constitutes multiplications of those terms.

Applying this representation to a task is simply applying the function to an initial *empty* meaning vector h_0 , which results in a transformed, *final* meaning vector h that then is used to make a decision on the phrase s . In the case of sentiment detection, a sentiment score $y(s)$ can be assigned to s as follows:

$$y(s) = h^\top u = h_0^\top M(s) u \quad (4.2)$$

In such a supervised task, matrix-space model parameters $\{M_w\}_{w \in \mathcal{V}}, h_0, u$ are learned from data. h_0 and u can be fixed (without reducing the representative

power of the model) to reduce the degrees of freedom during training.

4.3 Composition with Multiplicative Recurrent Neural Networks

Recurrent neural networks were defined in Chapter 3.

A property of recurrent neural networks is that input layer activations and the hidden layer activations of the previous time step interact additively to make up the activations for hidden layers at the current time step. This might be rather restrictive for some applications, or difficult to learn for modeling more complex input interactions. On the other hand, a multiplicative interaction of those layers might provide a better representation for some semantic analysis tasks. For sentiment detection, for example, “not” might be considered as a *negation* of the sentiment that comes after it, which might be more effectively modeled with multiplicative interactions. To this end, we investigate the multiplicative recurrent neural network (or the recurrent neural tensor network) for the sentiment analysis task that is the main focus of this paper (Socher et al., 2013).

Multiplicative recurrent networks retain the same interpretation of memory as Elman-type recurrent networks, the only difference being the recursive definition of h :

$$h_t = \sigma(x_t^\top A^{[1..|h|]} h_{t-1} + Wx_t + Vh_{t-1} + b) \quad (4.3)$$

$$y_t = \gamma(Uh_t + c) \quad (4.4)$$

where A is a $|h| \times |x| \times |h|$ tensor, and the bilinear operation $x^\top A y$ defines another vector as $(x^\top A y)_i = x^\top A^{[i]} y$ where the right-hand side represents the standard vector matrix multiplications and $A^{[i]}$ is a single slice (matrix) of the tensor A . This means that a single entry of $h_{t,i}$ is not only a linear combination of entries $x_{t,j}$ and $h_{t-1,k}$, but also includes multiplicative terms in the form of $a_{jk}^i x_{t,j} h_{t-1,k}$.

We can simplify Equation 4.3 and 4.4 by adding bias units to x and h :

$$h_t = \sigma(x_t'^\top A'^{[1..|h|]} h_{t-1}') \quad (4.5)$$

$$y_t = \gamma(U' h_t') \quad (4.6)$$

where $x' = [x; 1]$ and $h' = [h; 1]$. With this notation, W , V and b become part of the tensor A' and c becomes part of the matrix U' .

4.3.1 Ordinal regression with neural networks

Since fine-grained sentiment labels denote intensity in addition to polarity, our class labels are ordinal in nature. Therefore, we use an ordinal regression scheme for neural networks, as described in Cheng et al. (2008). Intuitively, each sentiment class denotes a threshold for which the instances belonging to the class have sentiment values less than or equal to. If an instance s belongs to class k , it automatically belongs to the lower order classes $1, \dots, k-1$, as well. Therefore, the target vector for instance s is $r = [1, \dots, 1, 0, \dots, 0]^\top$ where $r_i = 1$ if $i < k$ and $r_i = 0$ otherwise. This way, we can consider the output vector as a cumulative probability distribution on classes.

Because of the way class labels are defined, output response is not subject to normalization. Therefore, output layer nonlinearity in this case is the elemen-

twice sigmoid function ($1/(1 + e^{-x_i})$) instead of the softmax function ($e^{x_i} / \sum_j e^{x_j}$) which is traditionally used for multiclass classification.

Note that with this scheme, output of the network is not necessarily consistent. To decode an output vector, we firstly binarize each entry, by assigning 0 if the entry is less than 0.5 and 1 otherwise, as in conventional binary classification. Then we simply start from the entry with the lowest index, and whenever we observe a 0, we assume all of the entries with higher indices are also 0, which ensures that the resulting target vector has the proper ordinal form. As an example, $[1, 0, 1, 0]^\top$ is mapped to $[1, 0, 0, 0]^\top$. Then finally, we assign the corresponding integer label.

4.3.2 Relationship to matrix-space model

In this section we will show the connection between mRNTs and the matrix-space model.

Let us assume a purely multiplicative mRNT, without the bias units in the input and hidden layers (equivalently, $W = V = b = 0$). In such an mRNT, we compute the hidden layer (memory) as follows:

$$h_t = \sigma(x_t^\top A h_{t-1}) \quad (4.7)$$

Furthermore, assume $\sigma = I$ is the identity mapping, rather than a nonlinearity function. We can view the tensor multiplication in two parts: A vector x_t multiplied by a tensor A , resulting in a matrix which we will denote as $M(w_t)$, to make the dependence of the resulting matrix on the word w_t explicit. Then the matrix-vector multiplication $M(w_t)h_{t-1}$ resulting in the vector h_t . Therefore, we

can write the same equation as:

$$h_t = (x_t^\top A)h_{t-1} = M(w_t)h_{t-1} \quad (4.8)$$

and unfolding the recursion, we have

$$h_t = M(w_t)M(w_{t-1}) \dots M(w_1)h_0 \quad (4.9)$$

If we are interested in a scalar response for the whole sequence, we apply the output layer to the hidden layer at the final time step:

$$y_T = u^\top h_T = u^\top M(w_T) \dots M(w_1)h_0 \quad (4.10)$$

which is exactly the matrix space model if individual $M(w_t)$ were to be associated with the matrices of their corresponding words (Equation 4.2). Therefore, we can view mRNTs as a simplification to matrix-space models in which we have a tensor A to extract a matrix for a word w from its associated word vector, rather than associating a matrix with every word. This can be viewed as learning a matrix-space model with parameter sharing. This reduces the number of parameters greatly: instead of having a matrix for every word in the vocabulary, we have a vector per word, and a tensor to extract matrices.

Another interpretation of this is the following: instead of learning an individual linear operator M_w per word as in matrix-space models, mRNT learns $|x|$ number of *base* linear operators. mRNT, then, represents each word as a weighted sum of these base operators (weights given by the word vector x). Note that if x is a one-hot vector representation of a word instead of a dense word embedding (which means $|x| = |\mathcal{V}|$), then we have $|\mathcal{V}|$ matrices as the base set of operators, and x simply *selects* one of these matrices, essentially falling back to an exact matrix-space model (see Figure 4.1). Therefore mRNTs provide a natural transition of the matrix-space model from a one-hot sparse word representation to a low dimensional dense word embedding.

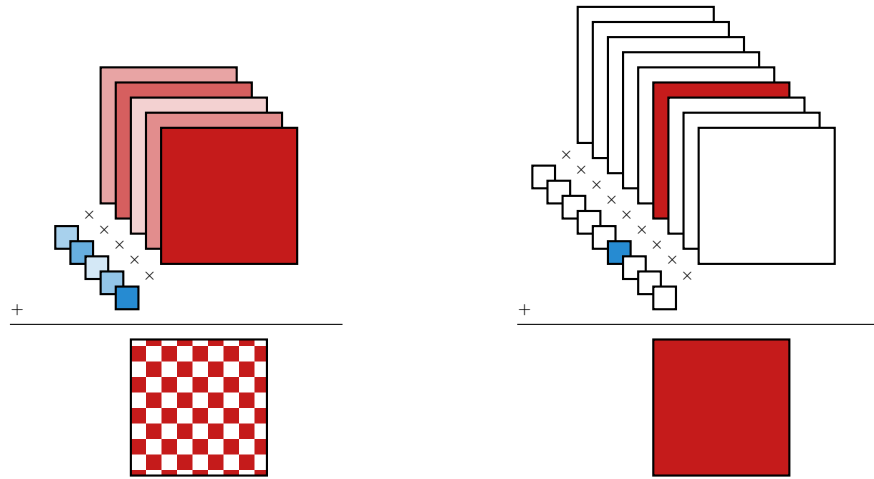


Figure 4.1: Vector x (blue) and tensor A (red) sliced along the dimension of x . **Left.** Dense word vector x computes a weighted sum over base matrices to get a square matrix, which then is used to transform the meaning vector. **Right.** One-hot word vector x with the same computation, which is equivalent to selecting one of the base matrices and falls back to a matrix-space model.

Besides a reduction in the number of parameters, another potential advantage of mRNTs over matrix-space models is that the matrix-space model is task-dependent: for each task, one has to learn one matrix per word in the whole vocabulary. On the other hand, mRNTs can make use of task-independent word vectors (which can be learned in an unsupervised manner) and only the parameters for the network would have to be task-dependent. This allows easier extension to multitask learning or transfer learning settings.

4.4 Experiments

4.4.1 Experimental Setting

Data. For experimental evaluation of the models, we use the manually annotated MPQA corpus (Wiebe et al., 2005) that contains 535 newswire documents annotated with phrase-level subjectivity and intensity. We use the same scheme as Yessenalina and Cardie (2011) to preprocess and extract individual phrases from the annotated documents, and convert the annotations to an integer ordinal label $\{0, 1, 2, 3, 4\}$ denoting a sentiment score from negative to positive. After preprocessing, we have 8022 phrases in total with an average length of 2.83. We use the training-validation-test set partitions provided by the authors to apply 10-fold CV and report average performance over ten folds.

Additionally, we use the Stanford Sentiment Treebank (SSTB) (Socher et al., 2013), which includes labels for 215,154 phrases in the parse trees of 11,855 sentences, with an average sentence length of 19.1. Similarly, real-valued sentiment labels are converted to an integer ordinal label in $\{0, \dots, 4\}$ by simple thresholding. We use the single training-validation-test set partition provided by the authors. We do not make use of the parse trees in the treebank since our approach is not structural; however, we include the phrase-level supervised labels (at the internal nodes of the parse trees) as labels for partial sentences.

Problem formulation. For experiments on the MPQA corpus, we employ an ordinal regression setting. For experiments on SSTB, we employ a simple multi-class classification setting, to make the models directly comparable to previous work.

In the classification setting, output nonlinearity γ is the softmax function, and the output y is a vector valued response with the class probabilities. Ordinal regression setting is as described in Section 4.3.1.

Evaluation metrics. For experiments using the MPQA corpus, we use the ranking loss as in Yessenalina and Cardie (2011), defined as $\frac{1}{n} \sum_i |y^i - r^i|$ where y and r are predicted and true scores respectively. For experiments using SSTB, we use accuracy, $\frac{1}{n} \sum_i \mathbb{1}(y^i = r^i)$ as in Socher et al. (2013).

Word vectors. We experiment with both randomly initialized word vectors (RAND) and pretrained word vector representations (VEC). For pretrained word vectors, we use publicly available 300 dimensional word vectors by Mikolov et al. (2013b), trained on part of Google News dataset (~ 100 B words). When using pretrained word vectors, we do not finetune them to reduce the degree of freedom of our models.

Additionally, matrix-space models are initialized with random matrices (RAND) or a bag-of-words regression model weights (BOW) as described in Yessenalina and Cardie (2011).

4.4.2 Results and Discussion

Quantitative results on the MPQA corpus are reported in Table 4.1. The top group shows previous results from Yessenalina and Cardie (2011) and the bottom group shows our results. Figure 4.2 shows reduced phrase representations (using PCA) for some phrases for RNT and mRNT trained on the MPQA corpus.

We observe that mRNT does slightly better than RNT with approximately the

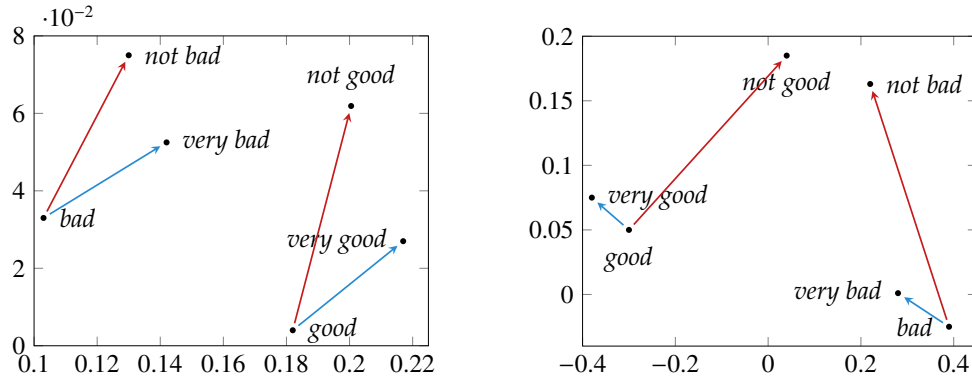


Figure 4.2: Hidden layer vectors reduced to 2 dimensions for various phrases. **Left.** Recurrent neural network. **Right.** Purely multiplicative recurrent neural tensor network. In mRNT, handling of negation is more nonlinear and correctly shifts the sentiment.

same number of parameters (0.5232 vs. 0.5265). This suggests that multiplicative interactions improve the model over additive interactions. Even though the difference is not significant in the test set, it is significant in the development set. We partially attribute this effect to the test set variance. This also suggests that multiplicative models are indeed more powerful, but require more careful regularization, because early stopping with a high model variance might tend to overfit to the development set.

The randomly initialized mRNT outperforms its equivalent randomly initialized matrix-space model (0.6799 vs. 0.7417), which suggests that more compact representations with shared parameters learned by mRNT indeed generalize better.

The mRNT and RNT that use pretrained word vectors get the best results, which suggests the importance of good pretraining schemes, especially when supervised data is limited. This is also confirmed by our preliminary experiments (which are not shown here) using other word vector training methods

Table 4.1: Average ranking losses (MPQA)

Method	Loss
PRank	0.7808
Bag-of-words LogReg	0.6665
Matrix-space _{Rand} ($d_h = 3$)	0.7417
Matrix-space _{BOW} ($d_h = 3$)	0.6375
$\text{RNT}_{\text{vec}}^+$ ($d_h = 315$)	0.5265
$\text{mRNT}_{\text{Rand}}^l$ ($d_h = 2$)	0.6799
$\text{mRNT}_{\text{vec}}^l$ ($d_h = 25$)	0.5278
$\text{mRNT}_{\text{vec}}^+$ ($d_h = 25$)	0.5232
$\text{mRNT}_{\text{vec}}^{\tanh}$ ($d_h = 25$)	0.5147

Table 4.2: Average accuracies (SSTB)

Method	Acc (%)
Bag-of-words NB	41.0
Bag-of-words SVM	40.7
Bigram NB	41.9
VecAvg	32.7
RSV^{\tanh}	43.2
MV-RSV^{\tanh}	44.4
mRSV^{\tanh}	45.7
$\text{RNT}_{\text{vec}}^+$ ($d_h = 315$)	43.1
$\text{mRNT}_{\text{vec}}^+$ ($d_h = 20$)	43.5

such as CW embeddings (Collobert and Weston, 2008) or HLBL (Mnih and Hinton, 2007), which yielded a significant difference (about 0.1 – 0.2) in ranking loss.

To test the effect of different nonlinearities, we experiment with the identity, rectifier and tanh functions with mRNTs. Experiments show that there is small but consistent improvement as we use rectifier or tanh over not using extra nonlinearity. The differences between rectifier and identity, and tanh and rectifier are not significant; however, the difference between tanh and identity is significant, suggesting a performance boost from using a nonlinear squashing function. Nonetheless, not using any nonlinearity is only marginally worse. A possible explanation is that since the squashing function is not the only source of nonlinearity in mRNTs (multiplicativeness is another source of nonlinearity), it is not as crucial.

Results on the Stanford Sentiment Treebank are shown in Table 4.2. Again, the top group shows baselines from Socher et al. (2013) and the bottom group shows our results.

Both RNT and mRNT outperform the conventional SVM and Naive Bayes baselines. We observe that RNT can get very close to the performance of Recursive Neural Network, which can be considered its structural counterpart. mRNT further improves over RNT and performs better than the recursive net and worse than the matrix-vector recursive net. Note that none of the recurrent models employ parse trees of sentences, unlike their recursive neural network variants.

4.5 Chapter Summary

In this chapter, we explored multiplicative recurrent neural networks as a model for the compositional interpretation of language. We evaluated them on the task of fine-grained sentiment analysis, in an ordinal regression setting and showed that mRNTs outperform previous work on MPQA, obtaining comparable results to previous work on Stanford Sentiment Treebank without using parse trees. We also described how mRNTs effectively generalize matrix-space models from a sparse one-hot word vector representation to a distributed, dense representation. This shows that these two approaches of sequential composition are actually tightly connected.

One benefit of mRNTs over matrix-space models is their separation of task-independent word representations (vectors) from task-dependent classifiers (tensor), making them very easy to extend for semi-supervised learning or

transfer learning settings. Slices of the tensor can be interpreted as *base matrices* of a simplified matrix-space model. Intuitively, every *meaning factor* (a dimension of the dense word vector) of a word has a separate operator acting on the meaning representation which we combine to get the operator of the word itself.

From a parameter sharing perspective, mRNTs provide better models. For matrix-space models, an update over a sentence affects only the word matrices that occur in that particular sentence. On the other hand, in an mRNT, an update over a sentence affects the global tensor as well. With such an update, the network alters its operation for similar words towards a similar direction.

One drawback of mRNTs over conventional Elman-type networks is their increased model variance, resulting from multiplicative interactions. This can be tackled by a stricter regularization.

CHAPTER 5

BIDIRECTIONAL RECURSIVE NEURAL NETWORKS FOR TOKEN-LEVEL LABELING WITH STRUCTURE

In previous chapters we have investigated sequential compositional models, now we focus on structural composition. In this chapter we propose bidirectional recursive neural networks to learn representations that exploit structural information and apply them to the task of opinion mining. The work described in this chapter is based on Irsoy and Cardie (2013).

As discussed in Chapter 1 and 2, we can exploit structure information when applying the composition function $f(\cdot, \cdot)$. This gives us a family of models which we consider as structural compositional models. Using the traditional perceptron layer for f results in a *recursive neural network* (Pollack, 1990). Given the structural representation of a sentence, e.g. a parse tree, recursive networks recursively generate parent representations in a bottom-up fashion, by combining tokens to produce representations for phrases, eventually producing the whole sentence (see Figure 5.1). The sentence-level representation (or, alternatively, its phrases) can then be used to make a final classification for a given input sentence — e.g. whether it conveys a positive or a negative sentiment.

Unfortunately, recursive neural networks generate representations only for the internal nodes in the structured representation. More formally, representation of every node is only a function of the subtree rooted at that node, therefore containing no information from outside of that subtree. This means that the leaf representations (words) completely lack any context information. As a result, they are not directly applicable to the token-level labeling tasks in which we are interested.

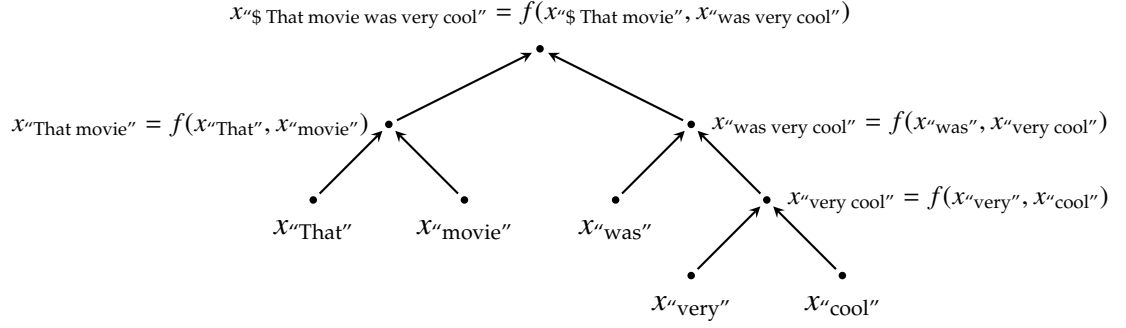


Figure 5.1: Structural recursive composition

To this end, in this chapter we propose and explore a neural network for token level tagging that aims to represent the structural information associated with a single token. In particular, we extend the traditional recursive neural network framework so that it not only generates representations for subtrees (i.e. phrases) and the whole sentence upward, but also propagates downward representations toward the leaves, carrying information about the structural environment of each word.

Our method is naturally applicable to any type of labeling task at the word level, however we limit ourselves to an opinion expression extraction task (same as in Chapter 3) in this work. In addition, although the method is applicable to any type of positional directed acyclic graph structure (e.g. the dependency parse of a sentence), we limit our attention in this initial study to binary parse trees (Socher et al., 2011b).

As described in Chapter 3, fine-grained opinion analysis aims to detect the subjective expressions in a text (e.g. “hate”) and to characterize their intensity (e.g. strong) and sentiment (e.g. negative) as well as to identify the opinion holder (the entity expressing the opinion) and the target, or topic, of the opinion

The	committee	,	as	usual	,	has
B_HOLDER	I_HOLDER	O	B_ESE	I_ESE	O	B_DSE
refused	to	make	any	statements	.	
I_DSE	I_DSE	I_DSE	I_DSE	I_DSE		O

Table 5.1: An example sentence with opinion expression, holder, and target labels

(i.e. what the opinion is about) (Wiebe et al., 2005).

In this chapter, in addition to the detection of *direct subjective expressions* (DSEs), *expressive subjective expressions* (ESEs) as in Chapter 3, we also focus on detecting opinion holders and targets, as defined in Wiebe et al. (2005). See Table 5.1 in contrast to Table 3.1.

5.1 Related Work

Recursive neural networks. Recursive neural networks operate on structured inputs which can be used to represent syntax in language (Pollack, 1990). They have been applied to parsing (Socher et al., 2011b), sentence-level sentiment analysis (Socher et al., 2011c), paraphrase detection (Socher et al., 2011a), morphology oriented word vector representation (Luong et al., 2013), question answering (Iyyer et al., 2014a) and political ideology detection (Iyyer et al., 2014b). We refer the reader to Section 2.3.3 for further reading on previous work in recursive neural networks.

Opinion mining. As described in Section 3.1, early work on fine-grained opinion extraction focused on recognizing subjective phrases (Wilson et al., 2005; Munson et al., 2005). Most approaches formulate the problem as a token-level sequence tagging problem with CRF-based approaches (Breck et al., 2007;

Choi et al., 2005; Choi and Cardie, 2010). Reranking approaches have been applied on top of sequence taggers (Johansson and Moschitti, 2010, 2011). More recent work relaxes the Markovian assumption of CRFs to capture phrase-level interactions Yang and Cardie (2012). Yang and Cardie (2013) propose a joint inference model to jointly detect opinion expressions, opinion holders and targets, and the relations among them, improving upon pipelined approaches.

5.2 Structural Composition with Recursive Neural Networks

Recursive neural networks (e.g. (Socher et al., 2011b)) comprise an architecture in which the same set of weights is recursively applied in a structural setting: given a positional directed acyclic graph, it visits the nodes in a topological order, and recursively applies transformations to generate further representations from previously computed representations of children. In fact, a recurrent neural network is simply a recursive neural network with a particular structure. Even though they can be applied to any positional directed acyclic graph, we limit our attention to recursive neural networks over positional binary trees, as in Socher et al. (2011b).

Given a binary tree structure with leaves having the initial representations, e.g. a parse tree with word vector representations at the leaves, a recursive neural network computes the representations at the internal node η as follows (see also Figure 5.2):

$$x_\eta = \sigma(W_L x_{l(\eta)} + W_R x_{r(\eta)} + b) \quad (5.1)$$

where $l(\eta)$ and $r(\eta)$ are the left and right children of η , W_L and W_R are the weight matrices that connect the left and right children to the parent, and b is a bias vec-

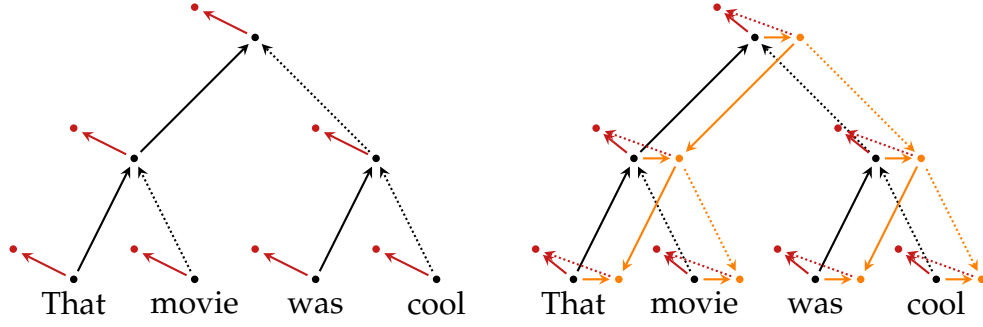


Figure 5.2: **Left.** Recursive neural network with bottom-up propagation. **Right.** Bidirectional recursive neural network with bottom-up and top-down propagation. Black, orange and red denote bottom-up, top-down and output layers, respectively. Connections that share the same color and style are shared.

tor. Given that W_L and W_R are square matrices, and not distinguishing whether $l(\eta)$ and $r(\eta)$ are leaf or internal nodes, this definition has an interesting interpretation: initial representations at the leaves and intermediate representation at the nonterminals lie in the same space. In the parse tree example, a recursive neural network combines representations of two subphrases to generate a representation for the larger phrase, in the same meaning space (Socher et al., 2011b). Depending on the task, we have a final output layer at the root ρ :

$$y = \gamma(Ux_\rho + c) \quad (5.2)$$

where U is the output weight matrix and c is the bias vector to the output layer. In a supervised task, supervision occurs at this layer. Thus, during learning, initial error is incurred on y , and backpropagated from the root, towards leaves (Goller and Kuchler, 1996).

5.3 Bidirectional Recursive Neural Networks

We will extend the aforementioned definition of a recursive neural network, so that it propagates information about the rest of the tree, to every leaf node, through structure. This will allow us to make decisions at the leaf nodes, with a summary of the surrounding structure.

First, we modify the notation in equation (5.1) so that it represents an upward layer through the tree:

$$x_\eta^\uparrow = \sigma(W_L^\uparrow x_{l(\eta)}^\uparrow + W_R^\uparrow x_{r(\eta)}^\uparrow + b^\uparrow) \quad (5.3)$$

Note that x_η^\uparrow is simply the initial representation x_η if η is a leaf, similar to equation (5.1). Next, we add a downward layer on top of this upward layer:

$$x_\eta^\downarrow = \begin{cases} \sigma(W_L^\downarrow x_{p(\eta)}^\downarrow + V^\downarrow x_\eta^\uparrow + b^\downarrow), & \text{if } \eta \text{ is a left child} \\ \sigma(W_R^\downarrow x_{p(\eta)}^\downarrow + V^\downarrow x_\eta^\uparrow + b^\downarrow), & \text{if } \eta \text{ is a right child} \\ \sigma(V^\downarrow x_\eta^\uparrow + b^\downarrow), & \text{if } \eta \text{ is root } (\eta = \rho) \end{cases} \quad (5.4)$$

where $p(\eta)$ is the parent of η , W_L^\downarrow and W_R^\downarrow are the weight matrices that connect the downward representations of parent to that of its left and right children, respectively, V^\downarrow is the weight matrix that connects the upward representation to the downward representation for any node, and b^\downarrow is a bias vector at the downward layer. A bidirectional recursive neural network is depicted in Figure 5.2.

Intuitively, for any node, x_η^\uparrow contains information about the subtree rooted at η , and x_η^\downarrow contains information about the rest of the tree, since every node in the tree has a contribution to the computation of x_η^\downarrow . Therefore x_η^\uparrow and x_η^\downarrow can be thought of as complete summaries of the structure around η . At the leaves (or

in general any node η , we use an output layer to make a final decision:

$$y_\eta = \gamma(U_\downarrow x_\eta^\downarrow + U_\uparrow x_\eta^\uparrow + c) \quad (5.5)$$

where U_\downarrow and U_\uparrow are the output weight matrices and c is the output bias vector.

In a supervised task, supervision occurs at the output layer. Then, during training, error backpropagates upwards, through the downward layer, and then downwards, through the upward layer, employing the backpropagation through structure method (Goller and Kuchler, 1996). If desired, backpropagated errors can be used to update the initial representation x , which allows the possibility of fine-tuning the word vector representations, in our setting.

Note that this definition is structurally similar to the unfolding recursive autoencoder (Socher et al., 2011a). However, the goals of the two architectures are different. The unfolding recursive autoencoder downward propagates representations as well, but the intention is to reconstruct the initial representations. We, on the other hand, want the downward representations x^\downarrow to be as different as possible from the upward representations x^\uparrow , since our aim is to capture the information about the rest of the tree rather than the particular subtree under investigation. Thus, the unfolding recursive autoencoder does not use x^\uparrow when computing x^\downarrow (except at the root), whereas our bidirectional recursive neural network does.

We will refer to the bidirectional recursive neural network as a BRSV neural network.

5.3.1 Incorporating Sequential Context

Depending on the task, one might want to employ the sequential context around each input vector as well, if the task has the sequential view in addition to structure. To this end, we can combine the bidirectional recurrent neural network with the bidirectional recursive neural network. This allows for the use of both the sequential information (past and future), and the structural information around a token to produce a final decision:

$$y_\eta = \gamma(U_{\rightarrow}h_\eta^{\rightarrow} + U_{\leftarrow}h_\eta^{\leftarrow} + U_{\downarrow}x_\eta^{\downarrow} + U_{\uparrow}x_\eta^{\uparrow} + c) \quad (5.6)$$

where h_η^{\rightarrow} and h_η^{\leftarrow} come from the sequential representation of the sentence, extracted from bidirectional recurrent neural networks.

This architecture can be seen as an extension to both the recurrent and the recursive neural network. During training, after the error term backpropagates through the output layer, individual errors per each of the combined architectures can be handled separately, which allows us to use the previously noted training methods per architecture.

We will refer to the aforementioned architecture as a COMBINED neural network.

5.4 Experiments

5.4.1 Experimental Setting

As in Chapter 3, we cast the problem of detecting DSEs and ESEs as two separate 3-class classification problems. We also experiment with joint detection of DSEs, opinion HOLDERS, and opinion TARGETs — as a 7-class classification problem with one Outside class and one Beginning and Inside class for DSEs, opinion holders and opinion targets. We compare the bidirectional recurrent neural network as described in Section 3.2.2 (BRNT), the bidirectional recursive network as described in Section 5.3 (BRSV), and the combined architecture as described in Section 5.3.1 (COMBINED). We use the Stanford PCFG parser to extract binary parse trees of sentences (Klein and Manning, 2003).

Evaluation Metrics. We use precision, recall and F-measure for performance evaluation. Since the boundaries of expressions are hard to define even for human annotators (Wiebe et al., 2005), we use two soft notions of the measures: *Binary Overlap* counts every overlapping match between a predicted and true expression as correct (Breck et al., 2007; Yang and Cardie, 2012), and *Proportional Overlap* imparts a partial correctness, proportional to the overlapping amount, to each match (Johansson and Moschitti, 2010; Yang and Cardie, 2012).

Data. We use the manually annotated MPQA corpus (Wiebe et al., 2005), which has 14492 sentences in total. For DSE and ESE detection, we separate 4492 sentences as a test set in which we evaluate the final performance, and run 10-fold cross validation for model selection. For joint detection of opinion HOLDER, DSE and TARGET, we have 9471 manually annotated sentences, and

we separate 2471 as a test set, and run 10-fold cross validation. A validation set is used during cross validation to pick the best regularization hyperparameter, simply a coefficient that penalizes the L2 norm.

Network Training. We use standard stochastic gradient descent, updating weights after minibatches of 80 sentences. We run 200 epochs for training. Furthermore, we fix the learning rate for every architecture, instead of tuning with cross validation, since initial experiments showed that in this setting, every architecture successfully converges without any oscillatory behavior.

Word Vector Representations. As initial representations of tokens, we use pre-trained Collobert-Weston embeddings (Collobert and Weston, 2008). Initial experiments with fine tuning the word vector representations presented severe overfitting, hence, we keep the word vectors fixed in the experiments.

Activation Units. We employ the standard softmax activation for the output layer: $\gamma(x) = e^{x_i} / \sum_j e^{x_j}$. For the hidden layers we use the rectifier linear activation: $\sigma(x) = \max\{0, x\}$. Experimentally, rectifier activation gives better performance, faster convergence, and sparse representations. Note that in the recursive network, we apply the same transformation to both the leaf nodes and the internal nodes, with the interpretation that they belong in the same meaning space. Employing the rectifier units at the upward layer causes the upward representations at the internal nodes to be always nonnegative and sparse, whereas the initial representations are dense, and might have negative values, which causes a conflict. To test the impact of this, we experimented with the sigmoid activation at the upward layer and the rectifier activation at the downward layer, which caused a degradation in performance. Therefore, at a loss of interpretation, we use the rectifier activation at both layers in our experiments.

Topology. The number of hidden layers per architecture is chosen so that every architecture to be compared has the same number of hidden units connected to the output layer as well as the same input dimensionality.

5.4.2 Results and Discussion

Model	Topology	Proportional			Binary		
		Prec.	Recall	F1	Prec.	Recall	F1
bRNT	(50, 75, 75)	56.59*	56.60*	56.60*	58.84	62.23	60.49
bRSV	(50, 150)	53.93	55.05	54.48	58.21	62.29	60.23
Combined	(50, 50, 50, 50)	54.22	53.25	53.73	58.59	62.72	60.59

Table 5.2: Experimental results for DSE detection

Model	Topology	Proportional			Binary		
		Prec.	Recall	F1	Prec.	Recall	F1
bRNT	(50, 75, 75)	45.69	53.72	49.38	52.13	65.43	58.03
bRSV	(50, 150)	42.64	53.49	47.45	47.15	71.19*	56.73
Combined	(50, 50, 50, 50)	46.16*	53.33	49.49	51.95	67.49	58.71*

Table 5.3: Experimental results for ESE detection

Model	Topology	DSE F1		Holder F1		Target F1	
		Prop.	Bin.	Prop.	Bin.	Prop	Bin.
bRNT	(50, 75, 75)	49.73	54.49	48.19	51.36	39.32*	50.53
Combined	(50, 50, 50, 50)	50.04	54.88	49.06*	52.20*	38.58	49.77

Table 5.4: Experimental results for joint HOLDER+DSE+TARGET detection

Experimental results for DSE and ESE detection are given in Tables 5.4.2 and 5.4.2. For the bRNT network, the topology (d_1, d_2, d_3) means that it has input dimensionality d_1 , forward hidden layer dimensionality d_2 , and backward dimensionality d_3 . For the bRSV network, (d_1, d_2) means that it has input dimen-

sionality and upward layer dimensionality d_1 and a downward layer dimensionality d_2 . For the COMBINED network, (d_1, d_2, d_3, d_4) means an input and upward layer dimensionality d_1 , downward layer dimensionality d_2 and forward and backward layer dimensionalities d_3 and d_4 . Asterisk (*) indicates that the performance is statistically significantly better than others in the group, with respect to a two-sided paired t-test with $\alpha = 0.05$.

We observe that the BRNT network has better performance than both the BRSV and the COMBINED architectures on the task of DSE detection, with respect to the proportional overlap metrics (56.60 F-measure, compared to 54.48, and 53.73). We do not observe a significant difference with respect to the binary overlap metrics. This is likely explained by the fact that DSEs tend to be shorter (often even a single word, such as “criticized” or “agrees”). Furthermore, since DSEs exhibit explicit subjectivity, they do not necessarily require a contextual investigation around the phrase. Most of the time, a DSE can be detected just by looking at the particular phrase.

On the task of ESE detection, the COMBINED network has statistically significantly better binary F-measure compared to others (58.71 compared to 58.03 and 56.73). Furthermore, the COMBINED network has statistically significantly better proportional precision than the two other architectures, at an insignificant loss in proportional recall. In terms of binary measures, the BRSV network has low precision and high recall, which might suggest a complementary behavior for the two architectures. ESEs tend to be longer relative to DSEs, which might explain the results. Additionally, unlike DSEs, ESEs more often require contextual information for their interpretation. For instance, in the given example in Table 5.1, it is not clear that “as usual” should be labeled as an ESE, unless one

looks at the context presented in the sentence.

Experimental results for joint detection of opinion HOLDER, DSE and TARGET, are given in Table 5.4.2 (not to be compared with Table 5.4.2, since the datasets are different). Here, the COMBINED architecture has insignificantly better performance in detecting DSEs (50.04 and 54.88 proportional and binary F-measures, compared to 49.73 and 54.49), and significantly better performance in detecting opinion HOLDERS (49.06 and 52.20 proportional and binary F-measures, compared to 48.19 and 51.36), whereas the BRNT network is better in detecting TARGETs (39.32 and 50.53 proportional and binary F-measures, compared to 38.58 and 49.77). Again, a possible explanation might be a better utilization of contextual information. To decide whether a named entity is an opinion holder or not, one must link (or fail to link) the entity to an opinion expression. Therefore, it is not possible to decide just by looking at the particular named entity.

For the joint detection task, we also investigate the performance on a subset of sentences, such that each sentence has at least one DSE and opinion holder, and they are separated by some distance. This is an attempt to explore the impact of the token-level sequential distance between an opinion holder and an opinion expression. The results are given in Figure 5.3. As the separation distance increases, on average, DSE detection performance of the combined architecture is steady for the COMBINED network compared to the BRNT network. This might suggest that structural information helps to better capture the cues between opinion HOLDERS and expression. Note that each distance-based subset of instances is strictly smaller, since there are fewer sentences conforming to the constraints, which causes an increase in variance.

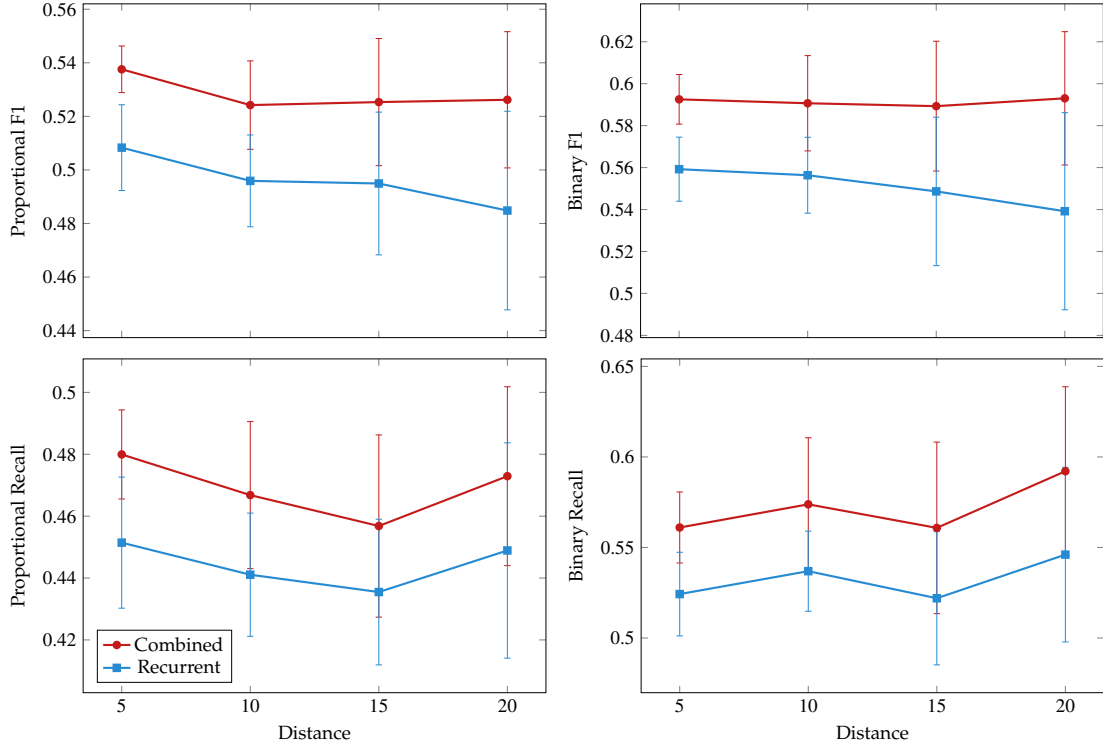


Figure 5.3: Experimental results for joint detection over sentences with separation

5.5 Chapter Summary

We have proposed an extension to the recursive neural network to carry out labeling tasks at the token level. We investigated its performance on the opinion expression extraction task. Experiments showed that, depending on the task, employing the structural information around a token might contribute to the performance.

In our bidirectional recursive neural network, the downward layer is built on top of the upward layer, whereas in the bidirectional recurrent neural network, forward and backward layers are separate. This causes the supervision to occur at a higher level in the recursive network relative to the recurrent net-

work, which makes training relatively more difficult. To alleviate this difficulty, an unsupervised pre-training of the upward layer, or a similar semi-supervised training, as in Socher et al. (2011c), might be employed. A fine tuning of the word vector representations during this pre-training might have a positive impact on the performance of the recursive network, since the learned representations for phrases might be structurally more meaningful, compared to the representations learned by sequential, or context window based approaches.

CHAPTER 6

DEEP RECURSIVE NEURAL NETWORKS FOR COMPOSITIONALITY IN LANGUAGE

In this chapter we propose deep recursive neural networks to employ feature hierarchies and feature re-use in structural neural compositional models, and present an application to fine-grained sentiment analysis. The work presented in this chapter is based on Irsoy and Cardie (2014a).

As discussed in the previous chapter, *recursive neural networks*, comprise a class of architecture that operates on structured inputs, and in particular, on directed acyclic graphs. A recursive neural network can be seen as a generalization of the recurrent neural network (Elman, 1990), which has a specific type of skewed tree structure (see Figure 6.1). Given the structural representation of a sentence, e.g. a parse tree, they recursively generate parent representations in a bottom-up fashion, by combining tokens to produce representations for phrases, eventually producing the whole sentence. The sentence-level representation (or, alternatively, its phrases) can then be used to make a final classification for a given input sentence — e.g. whether it conveys a positive or a negative sentiment.

Similar to how recurrent neural networks are deep in time, recursive neural networks are deep in structure, because of the repeated application of recursive connections. Recently, the notions of *depth in time* — the result of recurrent connections, and *depth in space* — the result of stacking multiple layers on top of one another, are distinguished for recurrent neural networks. In order to combine these concepts, *deep* recurrent networks were proposed (Schmidhuber, 1992; El Hihi and Bengio, 1995; Hermans and Schrauwen, 2013). They are con-

structured by stacking multiple recurrent layers on top of each other, which allows this extra notion of depth to be incorporated into temporal processing. Empirical investigations showed that this results in a natural hierarchy for how the information is processed (Hermans and Schrauwen, 2013). Inspired by these recent developments, we make a similar distinction between *depth in structure* and *depth in space*, and to combine these concepts, propose the *deep recursive neural network*, which is constructed by stacking multiple recursive layers.

The architecture we study in this work is essentially a deep feedforward neural network with an additional structural processing within each layer (see Figure 6.2). During forward propagation, information travels through the structure within each layer (because of the recursive nature of the network, weights regarding structural processing are shared). In addition, every node in the structure (i.e. in the parse tree) feeds its own hidden state to its counterpart in the next layer. This can be seen as a combination of feedforward and recursive nets. In a shallow recursive neural network, a single layer is responsible for learning a representation of composition that is both useful and sufficient for the final decision. In a deep recursive neural network, a layer can learn some parts of the composition to apply, and pass this intermediate representation to the next layer for further processing for the remaining parts of the overall composition.

To evaluate the performance of the architecture and make exploratory analyses, we apply deep recursive neural networks to the task of fine-grained sentiment detection on the recently published Stanford Sentiment Treebank (SSTB) (Socher et al., 2013).

We show that our deep recursive neural networks outperform shallow recursive nets of the same size in the fine-grained sentiment prediction task on the

Stanford Sentiment Treebank. Furthermore, our models outperform multiplicative recursive neural network variants, achieving new state-of-the-art performance on the task. We conduct qualitative experiments that suggest that each layer handles a different aspect of compositionality, and representations at each layer capture different notions of similarity.

6.1 Related Work

Recursive Neural Networks. We refer the reader to the previous chapter and Section 2.3.3 for a wide range of application areas of recursive networks in NLP.

Depth in Space vs Time. A key benefit of using deep (multi-stage) representation learners, for instance a deep feedforward neural network, is the feature hierarchy: Feature representations become more complex and abstract as we look at higher layers (Bengio, 2009). On the other hand, recursive application of the same weights (such as in recurrent neural networks) constrain each intermediate representation to lie in the same space, even though they are effectively still deep. Graves et al. (2013) make these two notions of depth explicit: Depth in space and depth in time (in the case of temporal processing). To take advantage of both hierarchical feature representations and variable length temporal processing, deep recurrent neural networks combine both of these notions by stacking multiple recurrent layers (Schmidhuber, 1992; El Hihi and Bengio, 1995; Graves et al., 2013; Hermans and Schrauwen, 2013). Our approach is similar, however instead of stacking temporal processing units, we stack structural layers that are more general and can be applied to arbitrary positional DAGs rather than only chains.

Sentiment Analysis. As mentioned in Chapter 4, sentiment analysis has been a very popular NLP task that can be defined at various granularities such as the word-, phrase-, sentence- or document-level (Pang and Lee, 2008). In addition to binary classification of polarity, i.e. positive or negative sentiment, fine-grained approaches focus on detecting intensity as well, e.g. positive or very positive (Yessenalina and Cardie, 2011; Socher et al., 2013). Stanford Sentiment Treebank dataset which contains sentences of movie reviews and their sentiment score with binary parse trees further popularized sentiment analysis as a task to examine structural compositional models (Socher et al., 2013). SSTB includes a supervised sentiment label for every node in the binary parse tree, not just at the root (sentence) level. This is especially useful for deep learning, since it allows a richer supervised error signal to be backpropagated across the network, potentially alleviating vanishing gradients associated with deep neural networks (Bengio et al., 1994).

6.2 Deep Recursive Composition

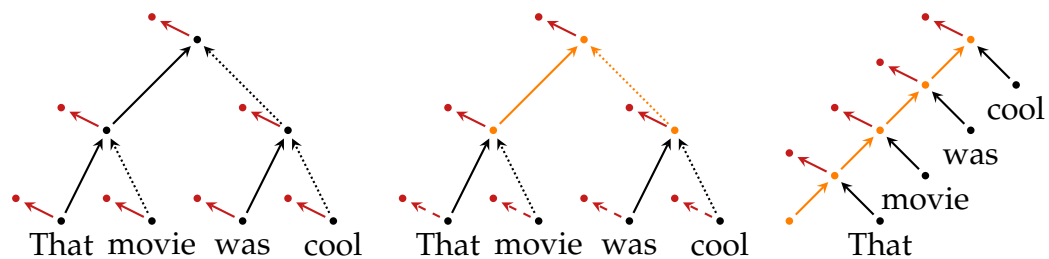


Figure 6.1: Operation of a recursive net (left), untied recursive net (middle) and a recurrent net (right) on an example sentence. Black, orange and red dots represent input, hidden and output layers, respectively. Directed edges having the same color-style combination denote shared connections.

Recursive neural networks are defined in Chapter 5.

6.2.1 Untying Leaves and Internals

Even though the previous definition of recursive neural networks, which treats the leaf nodes and internal nodes the same, has some attractive properties (such as mapping individual words and larger phrases into the same meaning space), in this work we use an untied variant that distinguishes between a leaf and an internal node. We do this by a simple parametrization of the weights W with respect to whether the incoming edge emanates from a leaf or an internal node (see Figure 6.1 (left) in contrast to (middle), color of the edges emanating from leaves and internal nodes are different):

$$h_\eta = \sigma(W_L^{l(\eta)} h_{l(\eta)} + W_R^{r(\eta)} h_{r(\eta)} + b) \quad (6.1)$$

where $h_\eta = x_\eta \in \mathcal{X}$ if η is a leaf and $h_\eta \in \mathcal{H}$ otherwise, and $W_\cdot^\eta = W_\cdot^{xh}$ if η is a leaf and $W_\cdot^\eta = W_\cdot^{hh}$ otherwise. \mathcal{X} and \mathcal{H} are vector spaces of words and phrases, respectively. The weights W_\cdot^{xh} act as a transformation from word space to phrase space, and W_\cdot^{hh} as a transformation from phrase space to itself.

With this untying, a recursive network becomes a generalization of the Elman type recurrent neural network with h being analogous to the hidden layer of the recurrent network (memory) and x being analogous to the input layer (see Figure 6.1 (right)). Benefits of this untying are twofold: (1) Now the weight matrices W_\cdot^{xh} , and W_\cdot^{hh} are of size $|h| \times |x|$ and $|h| \times |h|$ which means that we can use large pretrained word vectors and a small number of hidden units without a quadratic dependence on the word vector dimensionality $|x|$. Therefore, small but powerful models can be trained by using pretrained word vectors with a large dimensionality. (2) Since words and phrases are represented in different spaces, we can use rectifier activation units for σ , which have previously been shown to yield good results when training deep neural networks (Glorot et al.,

2011). Word vectors are dense and generally have positive and negative entries whereas rectifier activation causes the resulting intermediate vectors to be sparse and nonnegative. Thus, when leaves and internals are represented in the same space, a discrepancy arises, and the same weight matrix is applied to both leaves and internal nodes and is expected to handle both sparse and dense cases, which might be difficult. Therefore separating leaves and internal nodes allows the use of rectifiers in a more natural manner.

6.2.2 Deep Recursive Neural Networks

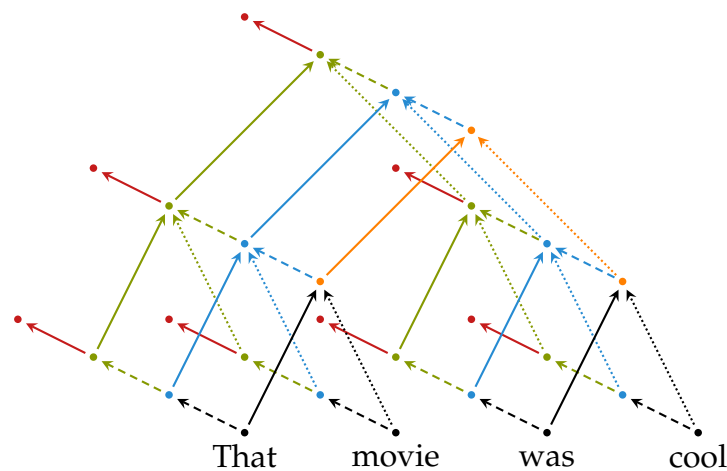


Figure 6.2: Operation of a 3-layer **deep recursive neural network**. Red and black points denote output and input vectors, respectively; other colors denote intermediate memory representations. Connections denoted by the same color-style combination are shared (i.e. share the same set of weights).

Recursive neural networks are deep in structure: with the recursive application of the nonlinear information processing they become as deep as the depth of the tree (or in general, DAG). However, this notion of depth is unlikely to involve a hierarchical interpretation of the data. By applying the same computation recursively to compute the contribution of children to their parents, and

the same computation to produce an output response, we are, in fact, representing every internal node (phrase) in the same space (Socher et al., 2011b, 2013). However, in the more conventional *stacked* deep learners (e.g. deep feedforward nets), an important benefit of depth is the hierarchy among hidden representations: every hidden layer conceptually lies in a different representation space and potentially is a more abstract representation of the input than the previous layer (Bengio, 2009).

To address these observations, we propose the *deep* recursive neural network, which is constructed by stacking multiple layers of individual recursive nets:

$$h_{\eta}^{(i)} = \sigma(W_L^{(i)} h_{l(\eta)}^{(i)} + W_R^{(i)} h_{r(\eta)}^{(i)} + V^{(i)} h_{\eta}^{(i-1)} + b^{(i)}) \quad (6.2)$$

where i indexes the multiple stacked layers, $W_L^{(i)}$, $W_R^{(i)}$, and $b^{(i)}$ are defined as before within each layer i , and $V^{(i)}$ is the weight matrix that connects the $(i-1)$ th hidden layer to the i th hidden layer.

Note that the untying that we described in Section 6.2.1 is only necessary for the first layer, since we can map both $x \in \mathcal{X}$ and $h^{(1)} \in \mathcal{H}^{(1)}$ in the first layer to $h^{(2)} \in \mathcal{H}^{(2)}$ in the second layer using separate $V^{(2)}$ for leaves and internals ($V^{xh(2)}$ and $V^{hh(2)}$). Therefore every node is represented in the same space at layers above the first, regardless of their “leafness”. Figure 6.2 provides a visualization of weights that are untied or shared.

For prediction, we connect the output layer to only the final hidden layer:

$$y_{\eta} = \gamma(Uh_{\eta}^{(\ell)} + c) \quad (6.3)$$

where ℓ is the total number of layers. Intuitively, connecting the output layer to only the last hidden layer forces the network to represent enough high level

information at the final layer to support the supervised decision. Connecting the output layer to all hidden layers is another option; however, in that case multiple hidden layers can have synergistic effects on the output and make it more difficult to qualitatively analyze each layer.

Learning a deep recursive network can be conceptualized as interleaved applications of the conventional backpropagation across multiple layers, and backpropagation through structure within a single layer. During backpropagation a node η receives error terms from both its parent (through structure), and from its counterpart in the higher layer (through space). Then it further backpropagates that error signal to both of its children, as well as to its counterpart in the lower layer.

6.3 Experiments

6.3.1 Experimental Setting

Data. For experimental evaluation of our models, we use the Stanford Sentiment Treebank (SSTB) (Socher et al., 2013), which includes labels for 215,154 phrases in the parse trees of 11,855 sentences, with an average sentence length of 19.1 tokens. Real-valued sentiment labels are converted to an integer ordinal label in $\{0, \dots, 4\}$ by simple thresholding. Therefore the supervised task is posed as a 5-class classification problem. We use the single training-validation-test set partitioning provided by the authors.

Baselines. In addition to experimenting among deep RSVs of varying width and depth, we compare our models to previous work on the same data. We use baselines from (Socher et al., 2013): a naive bayes classifier that operates on bigram counts (BINB), shallow recursive network (RSV) (Socher et al., 2011b,c) that learns the word vectors from the supervised data and uses tanh units, in contrast to our shallow RSVs, a matrix-vector RSV in which every word is assigned a matrix-vector pair instead of a vector, and composition is defined with matrix-vector multiplications (MV-RSV) (Socher et al., 2012b), and the multiplicative recursive net (or the recursive neural tensor network) in which the composition is defined as a bilinear tensor product (MRSV) (Socher et al., 2013). Additionally, we use a method that is capable of generating representations for larger pieces of text (PARAGRAPH VECTORS) (Le and Mikolov, 2014a), and the dynamic convolutional neural network (DCNN) (Kalchbrenner et al., 2014). We use the previously published results for comparison using the same training-development-test partitioning of the data.

Activation Units. For the output layer, we employ the standard softmax activation: $\gamma(x) = e^{x_i} / \sum_j e^{x_j}$. For the hidden layers we use the rectifier linear activation: $\sigma(x) = \max\{0, x\}$. Experimentally, rectifier activation gives better performance, faster convergence, and sparse representations. Previous work with rectifier units reported good results when training deep neural networks, with no pre-training step (Glorot et al., 2011).

Word Vectors. In all of our experiments, we keep the word vectors fixed and do not finetune for simplicity of our models. We use the publicly available 300 dimensional word vectors by (Mikolov et al., 2013b), trained on part of the

Google News dataset ($\sim 100\text{B}$ words).

Regularizer. For regularization of the networks, we use the recently proposed dropout technique, in which we randomly set entries of hidden representations to 0, with a probability called the dropout rate (Hinton et al., 2012b). Dropout rate is tuned over the development set out of $\{0, 0.1, 0.3, 0.5\}$. Dropout prevents learned features from co-adapting, and it has been reported to yield good results when training deep neural networks (Krizhevsky et al., 2012; Dahl et al., 2013). Note that dropped units are shared: for a single sentence and a layer, we drop the same units of the hidden layer at each node.

Since we are using a non-saturating activation function, intermediate representations are not bounded from above, hence, they can explode even with a strong regularization over the connections, which is confirmed by preliminary experiments. Therefore, for stability reasons, we use a small fixed additional L2 penalty (10^{-5}) over both the connection weights and the unit activations, which resolves the explosion problem.

Network Training. We use stochastic gradient descent with a fixed learning rate (.01). We use a diagonal variant of AdaGrad for parameter updates (Duchi et al., 2011). AdaGrad yields a smooth and fast convergence. Furthermore, it can be seen as a natural tuning of individual learning rates per each parameter. This is beneficial for our case since different layers have gradients at different scales because of the scale of non-saturating activations at each layer (grows bigger at higher layers). We update weights after minibatches of 20 sentences. We run 200 epochs for training. Recursive weights within a layer (W^{hh}) are initialized as $0.5I + \epsilon$ where I is the identity matrix and ϵ is a small uniformly random noise.

ℓ	$ h $	Fine-grained	Binary
1	50	46.1	85.3
2	45	48.0	85.5
3	40	43.1	83.5
1	340	48.1	86.4
2	242	48.3	86.4
3	200	49.5	86.7
4	174	49.8	86.6
5	157	49.0	85.5

Table 6.1: Results for RSVs. ℓ and $|h|$ denote the depth and width of the networks, respectively.

This means that initially, the representation of each node is approximately the mean of its two children. All other weights are initialized as ϵ . We experiment with networks of various sizes, however we have the same number of hidden units across multiple layers of a single RSV. When we increase the depth, we keep the overall number of parameters constant, therefore deeper networks become narrower. We do not employ a pre-training step; deep architectures are trained with the supervised error signal, even when the output layer is connected to only the final hidden layer. Additionally, we employ early stopping: out of all iterations, the model with the best development set performance is picked as the final model to be evaluated.

6.3.2 Results and Discussion

Quantitative Evaluation. We evaluate on both fine-grained sentiment score prediction (5-class classification) and binary (positive-negative) classification. For binary classification, we do not train a separate network, we use the network trained for fine-grained prediction, and then decode the 5 dimensional posterior probability vector into a binary decision which also effectively dis-

Method	Fine-grained	Binary
Bigram NB	41.9	83.1
RSV	43.2	82.4
MV-RSV	44.4	82.9
mRSV	45.7	85.4
DCNN	48.5	86.8
ParVec	48.7	87.8
dRSV (4, 174)	49.8	86.6

Table 6.2: Results for previous work and our best model (dRSV).

cards the neutral cases from the test set. This approach solves a harder problem. Therefore there might be room for improvement on binary results by separately training a binary classifier.

Experimental results of our models and previous work are given in Table 6.1 and 6.2. Table 6.1 shows our models with varying depth and width (while keeping the overall number of parameters constant within each group). ℓ denotes the depth and $|h|$ denotes the width of the networks (i.e. number of hidden units in a single hidden layer).

We observe that shallow RSVs get an improvement just by using pretrained word vectors, rectifiers, and dropout, compared to previous work (48.1 vs. 43.2 for the fine-grained task, see our shallow RSV with $|h| = 340$ in Table 6.1 and the RSV from Socher et al. (2013) in Table 6.2). This suggests a validation for untying leaves and internal nodes in the RSV as described in Section 6.2.1 and using pre-trained word vectors.

Results on RSVs of various depths and sizes show that deep RSVs outperform single layer RSVs with approximately the same number of parameters, which quantitatively validates the benefits of deep networks over shallow ones (see Table 6.1). We see a consistent improvement as we use deeper and narrower

networks until a certain depth. The 2-layer RSV for the smaller networks and 4-layer RSV for the larger networks give the best performance with respect to the fine-grained score. Increasing the depth further starts to cause a degrade. An explanation for this might be the decrease in width dominating the gains from an increased depth.

Furthermore, our best deep RSV outperforms previous work on both the fine-grained and binary prediction tasks, and outperforms Paragraph Vectors on the fine-grained score, achieving a new state-of-the-art (see Table 6.2).

We attribute an important contribution of the improvement to dropouts. In a preliminary experiment with simple L2 regularization, a 3-layer RSV with 200 hidden units each achieved a fine-grained score of 46.06 (not shown here), compared to our current score of 49.5 with the dropout regularizer.

Input Perturbation. In order to assess the scale at which different layers operate, we investigate the response of all layers to a perturbation in the input. A way of perturbing the input might be an addition of some noise, however with a large amount of noise, it is possible that the resulting noisy input vector is outside of the manifold of meaningful word vectors. Therefore, instead, we simply pick a word from the sentence that carries positive sentiment, and alter it to a set of words that have sentiment values shifting towards the negative direction.

In Figure 6.3, we give an example sentence, *“Roger Dodger is one of the best variations on this theme”* with its parse tree. We change the word *“best”* into the set of words *“coolest”, “good”, “average”, “bad”, “worst”*, and measure the response of this change along the path that connects the leaf to the root (labeled from 1 to 8). Note that all other nodes have the same representations, since

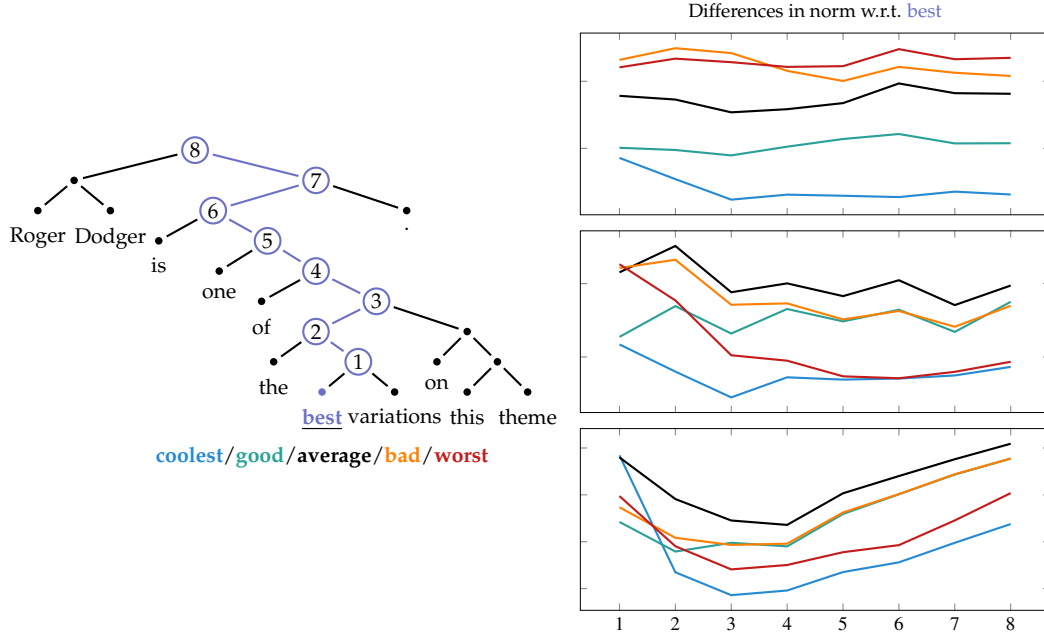


Figure 6.3: An example sentence with its parse tree (left) and the response measure of every layer (right) in a three-layered deep recursive net. We change the word “best” in the input to one of the words “coolest”, “good”, “average”, “bad”, “worst” (denoted by blue, light blue, black, orange and red, respectively) and measure the change of hidden layer representations in one-norm for every node in the path.

a node is completely determined by its subtree. For each node, the response is measured as the change of its hidden representation in one-norm, for each of the three layers in the network, with respect to the hidden representations using the original word (“best”).

In the first layer (bottom) we observe a shared trend change as we go up in the tree. Note that “good” and “bad” are almost on top of each other, which suggests that there is not necessarily enough information captured in the first layer yet to make the correct sentiment decision. In the second layer (middle) an interesting phenomenon occurs: Paths with “coolest” and “good” start close together, as well as “worst” and “bad”. However, as we move up in the tree, paths

with “*worst*” and “*coolest*” come closer together as well as the paths with “*good*” and “*bad*”. This suggests that the second layer remembers the *intensity* of the sentiment, rather than *direction*. The third layer (top) is the most consistent one as we traverse upward the tree, and correct sentiment decisions persist across the path.

charming results			
1	charming ,	interesting results	charming chemistry
2	charming and	riveting performances	perfect ingredients
3	appealingly manic and energetic	gripping performances	brilliantly played
4	refreshingly adult take on adultery	joyous documentary	perfect medium
5	unpretentious , sociologically pointed	an amazing slapstick instrument	engaging film
not great			
1	as great	nothing good	not very informative
2	a great	not compelling	not really funny
3	is great	only good	not quite satisfying
4	Is n't it great	too great	trashy fun
5	be great	completely numbing experience	fake fun

Table 6.3: Example shortest phrases and their nearest neighbors across three layers.

Nearest Neighbor Phrases. In order to evaluate the different notions of similarity in the meaning space captured by multiple layers, we look at nearest neighbors of short phrases. For a three layer deep recursive neural network we compute hidden representations for all phrases in our data. Then, for a given phrase, we find its nearest neighbor phrases across each layer, with the one-norm distance measure. Two examples are given in Table 6.3.

For the first layer, we observe that similarity is dominated by one of the words that is composed, i.e. “*charming*” for the phrase “*charming results*” (and “*appealing*”, “*refreshing*” for some neighbors), and “*great*” for the phrase “*not great*”. This effect is so strong that it even discards the negation for the second case, “*as great*” and “*is great*” are considered similar to “*not great*”.

In the second layer, we observe a more diverse set of phrases semantically. On the other hand, this layer seems to be taking syntactic similarity more into account: in the first example, the nearest neighbors of *“charming results”* are comprised of adjective-noun combinations that also exhibit some similarity in meaning (e.g. *“interesting results”*, *“riveting performances”*). The account is similar for *“not great”*: its nearest neighbors are adverb-adjective combinations in which the adjectives exhibit some semantic overlap (e.g. *“good”*, *“compelling”*). Sentiment is still not properly captured in this layer, however, as seen with the neighbor *“too great”* for the phrase *“not great”*.

In the third and final layer, we see a higher level of semantic similarity, in the sense that phrases are mostly related to one another in terms of sentiment. Note that since this is a supervised task on sentiment detection, it is sufficient for the network to capture only the sentiment (and how it is composed in context) in the last layer. Therefore, it should be expected to observe an even more diverse set of neighbors with only a sentiment connection.

6.4 Chapter Summary

In this chapter we proposed the deep recursive neural network, which is constructed by stacking multiple recursive layers on top of each other. We applied this architecture to the task of fine-grained sentiment classification using binary parse trees as the structure. We empirically evaluated our models against shallow recursive nets. Additionally, we compared with previous work on the task, including a multiplicative RSV and the more recent Paragraph Vectors method. Our experiments showed that deep models outperform their shallow coun-

terparts of the same size. Furthermore, deep RSV outperforms the baselines, achieving state-of-the-art performance on the task.

We further investigated our models qualitatively by performing input perturbation, and examining nearest neighboring phrases of given examples. These results suggest that adding depth to a recursive net is different from adding width. Each layer captures a different aspect of compositionality. Phrase representations focus on different aspects of meaning at each layer, as seen by nearest neighbor phrase examples.

CHAPTER 7

TOWARDS UNDERSTANDING NEURAL LANGUAGE LEARNERS

In previous chapters we have discussed various compositional neural architectures as feature learners to avoid feature engineering. However, it is still an open research problem to explain what features are learned by these feature learners. In this chapter we discuss challenges in understanding and interpreting neural compositional methods in NLP. Towards this goal we investigate several directions that might help researchers gain better insight.

Model interpretability is a key concept in machine learning and it may be essential in certain applications. Interpretable models allow humans to track the decisions made by the model. For example, in a decision tree, for a given test instance, a human can traverse the tree nodes and explain why the model decided in a certain way at each node (Perner, 2011). On the other hand, this might not be possible for black-box models such as neural networks, as discussed in Chapter 1. Neural networks implement a highly nonlinear function in a distributed manner, therefore the final decision function contains highly nonlinear interactions which make them very difficult to interpret. There are several reasons for why one might prefer an interpretable model over a black-box model:

- **Trust.** In certain domains *accountability* or *trust* is of importance when making decisions. For instance, in medical domains, doctors make a diagnostic decision which might be assisted by a machine learning method. In these cases, the doctor might need to understand the underlying factors for the decision to trust the model and go through with it.
- **Fixing systematic errors.** If researchers can explain the model behavior, this might lead to discovery of systematic errors made by the model which

can in turn be fixed and lead to better accuracy or performance.

- **Exploratory analysis.** In some applications, machine learning is not used to make predictions on unseen data but to make exploratory analysis and make sense of the data itself by discovering hidden structures and factors. In unsupervised learning, interpretable models can be used towards this end, and explanation of the model behavior would also explain the factors underlying the data to a human.
- **Research.** Model interpretation can be useful to researchers as well. If one can make sense of the behavioral changes caused by certain modifications to different architectures (for instance, adding a forget gate to the original LSTM formulation), this might lead to better informed architectural designs that realize desired behavior.

Compared to other application areas such as computer vision, interpreting neural models in the context of NLP has additional challenges brought by the extra temporal dimension, discreteness of the input space, and complexity of language in general. In this chapter we discuss some of these challenges and propose several simple tools for interpretation and visualization. The work presented here is not at all conclusive, however we intend it to serve a first step towards the end goal.

7.1 Related Work

In computer vision applications, visualization of neural network representations has been popular, possibly due to the nature of the task itself being visual. Even in simple feedforward neural networks, one can easily plot the norm

bounded activators of individual features (Ng, 2011). Applications of convolutional neural networks in the reverse direction have been explored to explain responsiveness of features in the input space (Zeiler and Fergus, 2014). Several previous works have shown that it is easy to mislead networks by imperceptible or non-interpretable changes to the input image (Szegedy et al., 2013; Nguyen et al., 2015), which further complicated model interpretation. Goodfellow et al. (2014) investigate the reasons for the existence of such adversarial examples and conclude that reasonable behavior occurs in a very thin manifold of the input space that includes the data.

Sequential neural models have an additional temporal dimension that brings extra complexity to the task. Hermans and Schrauwen (2013) investigate deep recurrent networks on the task of character level language modeling and show that stacking layers results in a temporal hierarchy: Higher layers capture longer term dependencies. Karpathy et al. (2015) utilize the same task to examine LSTMs and make an explicit error analysis by using various oracles to investigate what type of errors are being made by the models.

Specifically in the context of NLP, there is little work with the main goal of model understanding, even though many applications in NLP typically do some form of error analysis, inspection of nearest neighbors in representation spaces or visualizations through projections (e.g. Sutskever et al. (2014)). Li et al. (2015a) propose several simple visualization strategies that aim to interpret neural models in NLP, some of which we partially adopt.

7.2 Identifying Challenges

Our main goal is to investigate features automatically learned by a compositional neural model that is trained over natural language. In computer vision, a typical and simple way of understanding the feature represented by a single unit is to find the input that activates (*fires*) the unit the most, with respect to certain norm constraints (Ng, 2011). Intuitively, this shows the type of input that the unit wants to *detect*. This is done by casting it as an optimization problem in which the objective is the neuron activation, treating the input vector as the free parameter and keeping the model fixed, which can be solved by backpropagation into the input vector. Having a norm constraint ensures that the solution is meaningful: Otherwise any element of the input vector (e.g. a pixel) that has a positive weight can be increased indefinitely to improve the objective arbitrarily. This relates to the discussion of *input manifolds* which was given in the previous section; in a way, the additional norm constraint helps the vector to stay close to the input manifold.

Let us pose the same problem for a recurrent neural network that operates on natural language sentences (with the word vector projection layer). Note that in a recurrent network, a unit is a function of the entire prefix sequence, therefore our optimization is cast over all possible prefix sequences as the input to the network. Some of the challenges involved in this process is as follows:

1. **Discrete sequences.** In computer vision this optimization is easier due to having continuous input and fixed size, so backpropagation is possible. In language, input is variable-sized and discrete which makes optimization difficult. One can limit the length of a sentence and perform discrete



Figure 7.1: Feature activators in computer vision and in NLP. **Left.** Highest activating image patches of a hidden unit of a single layer perceptron autoencoder. We can easily see the common pattern being edge-like (actual values for the splitting hyperplane shown below). **Right.** Highest activating sentences of a hidden unit of a recurrent net.

of image patches and realizing that they are edges, however in language it might not be as simple, e.g. see Figure 7.1.

4. **Feature disentanglement.** Attempts to explain model behavior through hidden unit activations assume that features are captured individually in each unit, or inspecting each feature separately might provide enough insight. However there is no reason to expect uncorrelated feature activations. In fact, Szegedy et al. (2013) suggest that it is the space as a whole, rather than each individual unit, that contains the semantic information in the higher layers of neural networks.

- One possible approach to tackle this issue is to incorporate additional constraints that have disentangled features (distinct underlying factors of variation). Even though this has the potential to improve our understanding of neural models in general, it is difficult to say how much of that understanding might transfer to models trained without the additional constraints. Cirik et al. (2016) show that even changing the order in which data is given to the model during training can change the model behavior drastically.
- Another approach is to train a neural model without the constraints and then disentangle the features afterwards. Perhaps the simplest method is to apply principle component analysis (PCA) (Jolliffe, 2002) to decorrelate hidden features. Feature disentanglement is an active research area (Bengio et al., 2013).

7.3 Visualizing Hidden Features

To find **feature activators** we follow a very simplistic approach: given a trained model, we pass over the sentences in a given dataset and then sort the hidden unit activations to look at top K instances. This is akin to solving the above optimization problem over a finite (and tractable) number of candidate instances rather than an infinite input space. This approach addresses discreteness (1) and possibly the input manifold (2) issues mentioned in the previous section. If the original dataset that is used to train the model itself is small, one can use other, even unsupervised datasets to perform such an analysis. However one should take caution, a dataset of natural language sentences does not necessarily contain instances from the original input manifold since there might be differences

with respect to domain, content, or linguistic style. In fact, the manifold of natural language sentences is not necessarily a *hard set*, but likely a distribution over all possible sequences of words (e.g. given by a language model), thus, having a large enough dataset means eventually observing a rare sentence that can have very low probability. For instance, the example sentence given in the previous section ("*bark, ..., bark !*") comes from the Book Corpus (Zhu et al., 2015b).

For feature explanation (3), we simply perform manual inspection. Even though this is laborious and the conclusions that one can deduce are possibly limited, it can still be used to see if there are obvious and easily explainable features, and construct hypotheses in order to perform further inspection. One can envision other methods to explain the activators, for instance using an interpretable machine learning method to model the activator sentences, however we leave such approaches as future work.

In the first part of our analysis, for simplicity, we do not address feature disentanglement (4) at all. We then perform further analyses with decorrelated features using PCA.

In addition to investigating feature activators, we also inspect the *most divergent sentences* with respect to a given feature: For a hidden unit of interest, we clamp it to zero when we are performing a forward-pass and evaluate how much the output response y' changes with respect to the original output y without the clamping. By sorting with respect to $\|y' - y\|$, we have the instances that are most impacted by the removal of the feature. Compared to looking for feature activators, this analysis has the advantage that the activations are not necessarily saturated and can be in the linearly operating region which makes first-order statistics (which will be described later) more meaningful, whereas in

activator inspection we explicitly desire high activations, thus sigmoidal units are likely to be saturated and have derivatives close to zero.

In the following subsections we describe the activation statistics we evaluate and visualize (in addition to simply plotting the activations $h = \{h_t\}_{t=1}^T$ themselves).

7.3.1 First Order Saliency and Expected Deviation

Let us say that we are inspecting feature i that has high activation on some sentence $s = \{w_t\}_{t=1}^T$ that has the highest activation at time step t^* , i.e. $t^* = \operatorname{argmax}_t \{h_{t,i}\}$. Let x_w denote the word vector for a word $w \in \mathcal{V}$, and as an abuse of notation, set $x_t = x_{w_t}$ for $t \in \{1, \dots, T\}$ when not ambiguous.

A question of interest is which words in the sentence had the most impact in this activation, which might better explain the purpose of the feature. A statistic that we can use towards this end is the **first-derivative saliency**:

$$|\delta x_t| = \left| \frac{\partial h_{t^*,i}}{\partial x_t} \right| \quad (7.1)$$

for all $t \leq t^*$, where $|\cdot|$ is the elementwise absolute value. This shows the (absolute value of) first order dependency of the activation to word t . Li et al. (2015a) apply this with respect to the final decision by taking the derivative of the final scoring function whereas we compute derivatives of hidden feature activations. Li et al. (2015a) also inspect every dimension of $|\delta x_t|$, whereas we summarize it as a single scalar value for each word using a norm, as $\|\delta x_t\|_{L1}$ or $\|\delta x_t\|_{L2}$.

Since this relates to the first order approximation of $h_{t^*,i}$ as a function of x_t , it is useful so long as the linear approximation is good enough. When the function

is highly nonlinear, this might not be necessarily true. Furthermore, when the activations are close to saturation, derivatives will be close to zero even though there might be a high dependence on a specific word. Because of these potential issues, we propose another statistic which we call **expected deviation**. This is not intended to replace but rather complement the first order analysis.

For a word w_t , we can measure its impact on $h_{t^*,i}$ by simply substituting it with other words and evaluating how much the activation changes on average:

$$\frac{1}{|\mathcal{V}|} \sum_{w'_t \in \mathcal{V}} |h_{t^*,i}(s) - h_{t^*,i}(s')| = \mathbb{E}_{w'_t \sim \mathcal{U}(\mathcal{V})}[|h_{t^*,i}(s) - h_{t^*,i}(s')|] \quad (7.2)$$

where s' is the sentence that has word w'_t substituted for w_t in s and the expectation is taken over the uniform distribution over the vocabulary, $\mathcal{U}(\mathcal{V})$. However, there will be many instances where s' is not a natural sentence (or even grammatical) and as we discussed, there is no need to expect reasonable behavior outside of the input manifold used to train the model and out-of-manifold values might potentially taint the statistic. Therefore, we use a substitute distribution \mathcal{S} (Yuret, 2012), that assigns probabilities to each substitute word given its context, $\mathbb{P}(w'_t | s - w_t)$, computed by a language model:

$$(w'_t | s - w_t) \sim \mathcal{S}(s - w_t) \quad (7.3)$$

Then the expectation is taken over the substitute distribution \mathcal{S} instead of the uniform distribution:

$$\sum_{w'_t \in \mathcal{V}} \mathbb{P}(w'_t | s - w_t) |h_{t^*,i}(s) - h_{t^*,i}(s')| = \mathbb{E}_{w'_t \sim \mathcal{S}(s - w_t)}[|h_{t^*,i}(s) - h_{t^*,i}(s')|] \quad (7.4)$$

We shorten the above expression as $\mathbb{E}[\Delta h_i]$ to avoid clutter.

For example, when a word is almost completely determined by its context, i.e. $\mathbb{P}(w'_t = w_t | s - w_t) \approx 1$, the expectation will return a value close to zero,

which can be more intuitive because the information contained in the word is already almost entirely contained in its context, and since it brings little new information, we consider that word to have very small impact. Using a uniform distribution will not have this behavior.

Expected deviation has the advantage that it is not affected by the structure of the word vector space (e.g. the shape of the manifold of natural word vectors in the continuous space) since it directly manipulates word indices; h is treated as a function of $\{w_t\}_t$ rather than $\{x_{w_t}\}_t$. Furthermore, it is not affected by vanishing gradients or hard saturation since it does not use derivatives. It has the drawbacks that it depends on the quality of the substitute model and is less efficient to compute.

7.3.2 Cross-Feature First Order Saliency

Another application of the first order method described above could be used to assess cross-feature dependencies. Instead of taking the derivative of an activation $h_{t^*,i}$ of interest with respect to x_t , we can take it with respect to h_t for $t < t^*$ to evaluate (first order) temporal dependencies between $h_{t^*,i}$ and $h_{t,j}$. Thus, if we have a sense of what feature j implements, we might be able to use that information to interpret feature i .

Note that although the recurrent matrix (or a combination of them such as V , V^z , and V^r in GRUs) might be considered an approximation to saliency, cross-feature saliency is different than simply inspecting the recurrent matrix V : The recurrent matrix itself is not context dependent and does not provide any information about how the cross-feature dependencies change conditioned on the

current prefix sequence $\{w_t\}_{t < t^*}$.

7.3.3 Immediate Temporal Differences

Li et al. (2015a) use variance of word vectors as a simple measure of word importance for a given task, after training with word vectors as free parameters. More concretely, they measure the deviation of a word vector of interest, from the mean word vector where the mean is taken over the sentence. This is, in a way, a measure of how much the training process updated a word vector x compared to other word vectors in the sentence. This has the drawback of (1) not being applicable to the case where pretrained word vectors are fixed and (2) having little context information. So if a word is useful in general for a given task (e.g. a sentiment word for sentiment classification), it will have a high value regardless of being useful in that particular context or not. To illustrate how this might be an issue, imagine a sequence model for sentiment classification that starts with neutral decision and updates it to positive or negative as soon as it observes a sentiment word, and otherwise maintains its decision. For an example sentence such as “*great movie, great cinematography and great storytelling*”, variance analysis would assign the same importance to all occurrences of “*great*”. However the first occurrence had the most impact on the decision of the model.

Looking at h instead of x might possibly provide better context and positionality information, and can be applied to the case where word vectors are kept fixed. By a similar intuition, one can inspect deviation of a state vector h_t from the mean state vectors over the sentence. However this might not be equally useful, since we might observe highly impactful words to alter the state but

then the state can remain roughly the same throughout the rest of the sentence to maintain relevant information, which would lead to very small deviation of h_t with respect to the sentence average, despite the fact that w_t was important.

Alternatively we can plot $h_t - h_{t-1}$ (we use the shorthand $h - \bar{h}$ to avoid using the temporal subscript) to see how much of a jump word t caused in the state representation. We expect highly impactful words to cause bigger temporal jumps. Again, we can summarize this using a norm $\|h_t - h_{t-1}\|_{L2}$.

7.3.4 Fine-grained Activation Statistics

To provide more detailed information, in addition to plotting h_t , we can visualize different components of h . For instance, in an Elman-type recurrent network, where

$$h_t = \sigma(Wx_t + Vh_{t-1} + b) \quad (7.5)$$

in addition to inspecting $h_{t,i}$ for a feature i of interest, we can evaluate $(Wx_t)_i$ or $(Vh_{t-1})_i$ to assess how much impact the current word x_t or past history h_{t-1} had on this particular activation $h_{t,i}$. This would potentially help us understand whether a feature was just activated or modified, or it is a continuation of an already activated signal.

Depending on the architecture, there might be further components. With Gated Recurrent Units (defined in 7.4.1), we visualize update gates z and the candidate memory \tilde{h} in addition to h , as well as $(Wx_t)_i$, $(W^z x_t)_i$, $(W^r x_t)_i$ to assess the impact of the word and $(V(r_t \cdot h_{t-1}))_i$, $(V^z h_{t-1})_i$, $(V^r h_{t-1})_i$ for past history.

7.4 Experiments

7.4.1 Experimental Setting

Task. Ideally, we are interested in interpreting generic language learners such as a language model, or a skip-thought model (Kiros et al., 2015), to be able to understand representations that are useful for language processing in general. However to make feature inspection more tractable, we use fine-grained sentiment analysis as our task. Sentiment analysis is a good task in the sense that it is not trivially easy, yet is simple enough that a small sequential model can perform reasonably well. As before, we use the Stanford Sentiment Treebank (SSTB) (Socher et al., 2013), which has 11,855 sentences with fine-grained sentiment labels ($\{0, \dots, 4\}$) (we discard the parse trees and phrase level labels).

Our GRU model with 25 hidden units achieves 46.83% 5-class accuracy, where current state-of-the-art is around 50%. On the other hand, an unsupervised model such as a language model or a skip-thought model mentioned above, typically requires hundreds or thousands of hidden units (Mikolov et al., 2011; Kiros et al., 2015), which would make manual inspection of features intractable.

Architecture. Several popular neural architectures exist for sequence modeling as we have discussed in Chapter 1. We desire architectures that are simple enough so that interpretation is potentially easier, but also powerful enough that they are sufficiently accurate to be of interest. After preliminary experiments we have settled on the Gated Recurrent Unit (GRU) (Cho et al., 2014), which was more accurate than Elman-type recurrent networks and is simpler than LSTMs.

LSTMs represent state in two components, an internal representation c that is visible only to its own memory, and an external memory representation h , as a function of c , that is visible to external components (Hochreiter and Schmidhuber, 1997; Greff et al., 2015). Because of this formulation the composition function cannot be represented in terms of h and x as $h_t = f(h_{t-1}, x_t)$ as we have discussed in previous chapters, but rather as $[h_t; c_t] = f([h_{t-1}; c_{t-1}], x_t)$, which effectively doubles the state space dimensionality (given that $|c| = |h|$) and brings additional complexity.

GRU formulation is as follows:

$$z_t = \text{sigmoid}(W^z x_t + V^z h_{t-1} + b^z) \quad (7.6)$$

$$r_t = \text{sigmoid}(W^r x_t + V^r h_{t-1} + b^r) \quad (7.7)$$

$$\tilde{h}_t = \sigma(W x_t + V(r_t \cdot h_{t-1}) + b) \quad (7.8)$$

$$h_t = z_t \cdot \tilde{h}_t + (1 - z_t) \cdot h_{t-1} \quad (7.9)$$

$$y_t = \gamma(U h_t + c) \quad (7.10)$$

where $\mathbf{1}$ is the all-ones vector, \cdot is elementwise multiplication, and z and r are *update* and *reset* gates, respectively. Sigmoid is defined as (elementwise) $1/(1 + e^{-x})$. As before, $\sigma(\cdot)$ and $\gamma(\cdot)$ are hidden layer and output layer nonlinearities, respectively. In this instance we use tanh for the hidden layer nonlinearity and softmax for the output layer since we have a classification problem.

We use pretrained word vectors and do not fine-tune them to reduce the total number of parameters of the model. For pretrained word vectors, we use the publicly available 300 dimensional word vectors by Mikolov et al. (2013b), trained on part of Google News dataset (~ 100 B words).

Substitute model for measuring expected deviation. We trained a 4-gram

LM using SRILM (Stolcke, 2002) with interpolated Kneser-Ney discounting. We used ukWaC corpora for English (Ferraresi et al., 2008). Words observed fewer than two times were replaced with an ‘unknown word’ tag. We used the FAST-SUBS algorithm (Yuret, 2012) to generate the top 100 substitute words and their substitute probabilities.

Unlabeled data. In some instances, to search for activators from a larger input space, we use an unlabeled dataset larger than SSTB. We use the sentences from the unprocessed version of the movie review dataset from Pang et al. (2002), since it contains sentences about movie reviews and matches the domain of SSTB. This data contains roughly 900k sentences.

7.4.2 Results and Discussion

Movie keywords. The top activating sentences for feature 4 (of 25) are shown in Table 7.1, and some sentences are visualized in Figures 7.2 and 7.3. At a first glance, this feature seems to respond to keywords about movies, such as *movie*, *cinematography*, or *screenplay*. Interestingly, it has an output vector (i.e. 4th column of U) of $[0.13, -0.21, -0.19, 0.00, 0.76]^\top$, which means it assigns positive scores to the labels of **very positive** and **very negative**. Note the consistent behavior of $(V\bar{h})_4$ and z_4 in some of the examples: After encountering a relevant keyword, the model tends to assign low values to the update gates (most consistent negative vote comes from the history) and maintain that information.

When we look at which words contribute to activation of this feature, we make a similar observation. In Table 7.2, we inspect the top activating word vectors of input layer contributions to the activation, $(Wx)_4$ and $(W^z)_4$. We see

great performances , stylish cinematography and a gritty feel help make gangster no. 1 a worthwhile moviegoing experience .
everyone should be able to appreciate the wonderful cinematography and naturalistic acting .
with danilo donati 's witty designs and dante spinotti 's luscious cinematography , this might have made a decent children 's movie – if only benigni had n't insisted on casting himself in the title role .
first and foremost ... the reason to go see “ blue crush ” is the phenomenal , water-born cinematography by david hennings .
good fun , good action , good acting , good dialogue , good pace , good cinematography .
greengrass has delivered an undoubted stylistic tour-de-force , and has managed elements such as sound and cinematography with skill
one of the best films of the year with its exquisite acting , inventive screenplay , mesmerizing music , and many inimitable scenes of tenderness , loss , discontent , and yearning .
one of creepiest , scariest movies to come along in a long , long time , easily rivaling blair witch or the others .
with this masterful , flawless film , -lrb- wang -rrb- emerges in the front ranks of china 's now numerous , world-renowned filmmakers .
apart from dazzling cinematography , we 've seen just about everything in blue crush in one form or the other .
this is dicaprio 's best performance in anything ever , and easily the most watchable film of the year .
oscar wilde 's masterpiece , the importance of being earnest , may be the best play of the 19th century .
insomnia is one of the year 's best films and pacino gives one of his most daring , and complicated , performances .
bubba ho-tep is a wonderful film with a bravura lead performance by bruce campbell that does n't deserve to leave the building until everyone is aware of it .
a beautifully made piece of unwatchable drivel .
gorgeous scenes , masterful performances , but the sickly sweet gender normative narrative left an acrid test in this gourmet 's mouth .
this is one of the year 's best films .

Table 7.1: Top activators of feature 4. Maximum activating words are shown in boldface.

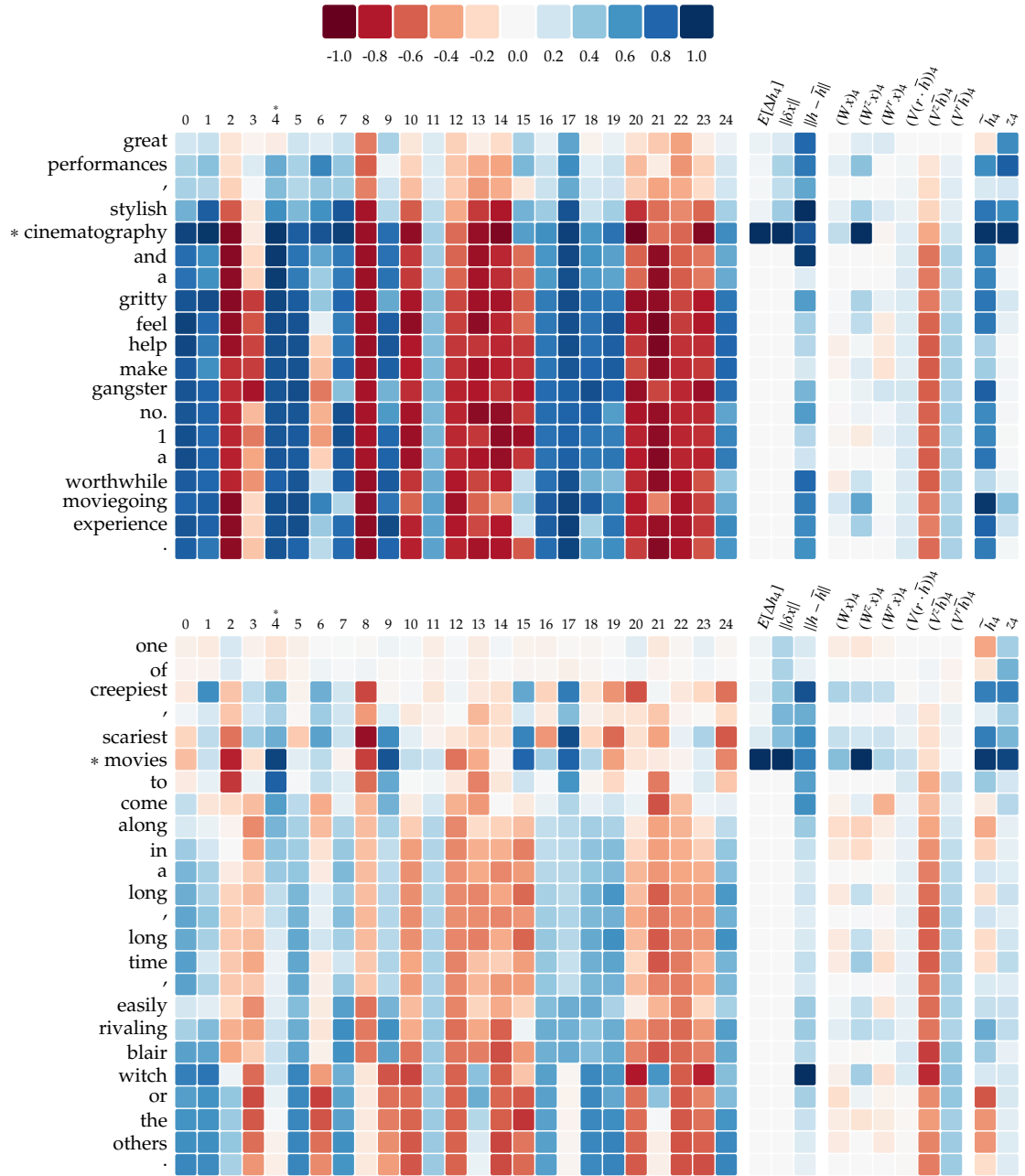


Figure 7.2: Some high activators of feature 4.

keywords about movies and sometimes words that have positive sentiment or connotation.

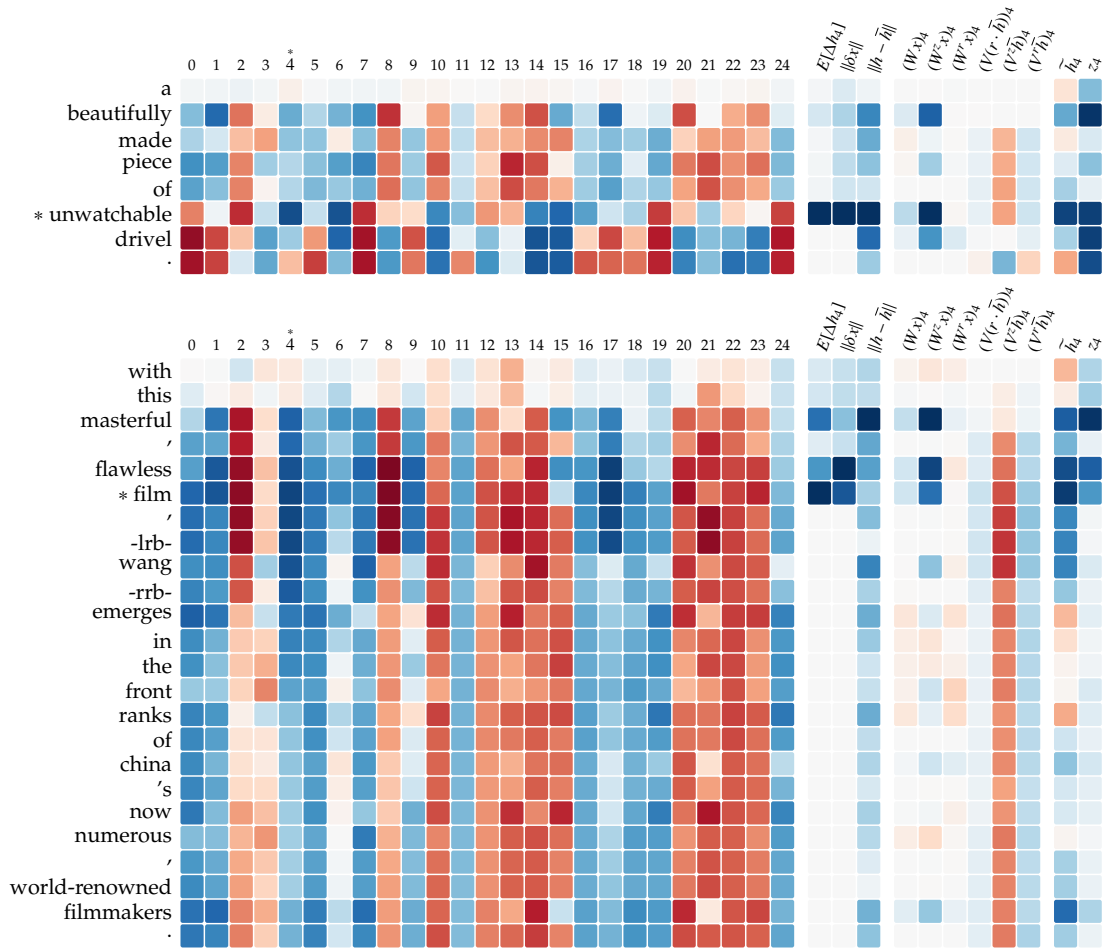


Figure 7.3: More high activators of feature 4.

“Most ... ever ...” When we inspect sample sentences that have high activations of feature 9, we observe syntactically similar phrases that carry positive sentiment such as *“most ... i ‘ve ever seen”* or *“... like nothing we ‘ve ever seen before”*. The top two examples are shown in Figure 7.4. Similar instances exist in high activators of unlabeled data as well. However the phrasing is not shared across all instances.

In fact, when we investigate the most impacted sentences when we force feature 9 to zero, we observe many activations in the negative direction. Most negative activators typically seem to contain negative sentiment words, such as

cinematography (1.63), blockbusters (1.63), overproduced (1.53), acclaim (1.52), oscar (1.51), camerawork (1.51), gore (1.5), moviegoers (1.42), sequels (1.41), cinematographer (1.4), grosses (1.39), comedies (1.38), movies (1.37), audiences (1.37), unwatchable (1.35), rapturous (1.34), masterpiece (1.34), megaplex (1.34), filmgoers (1.34), dreck (1.34), screens (1.33), wickedly (1.33), films (1.32), acclaimed (1.32), laurels (1.3), raving (1.3), lovably (1.3), remakes (1.28), nonstop (1.25), mesmerize (1.25), theatrically (1.23), artistically (1.23), moviegoing (1.23), theaters (1.23), deliriously (1.23), trilogy (1.22), screenings (1.21), hyped (1.2), watchable (1.2), visuals (1.2), cheesiest (1.17), coulda (1.17), masterful (1.17), campy (1.16), schlock (1.16), sequel (1.15), outdoes (1.15), bilked (1.14), classics (1.14), amaze (1.14)
--

cinematography (6.36), camerawork (6.22), curtsy (5.99), embalmed (5.65), overlong (5.59), overcook (5.58), stagy (5.49), diction (5.39), stilted (5.29), monologue (5.28), yorker (5.26), watchable (5.23), talky (5.19), costuming (5.14), cameo (5.07), unwatchable (5.02), cinematographer (5.01), novelistic (5), virtuosity (4.98), chiaroscuro (4.97), subtitles (4.97), miscast (4.95), lyricism (4.88), simmer (4.87), rawness (4.84), miniseries (4.83), glimpses (4.82), lensing (4.82), physique (4.81), biopic (4.77), memoir (4.76), masterful (4.76), bravura (4.75), fluidly (4.74), underdone (4.74), underplays (4.74), expressiveness (4.72), cinematically (4.72), outshine (4.7), admirably (4.69), choppiness (4.66), naturalness (4.65), narration (4.64), shadings (4.63), dulls (4.63), sensuality (4.62), entrancing (4.62), stylishness (4.61), engrossing (4.61), mesmerizing (4.6)

Table 7.2: Highest activating words of $(Wx)_4$ (top) and $(W^zx)_4$ (bottom). Activation amounts are given in parentheses.

unconvincing, *unfunny* or *unromantic*, sometimes accompanied with an adverb. Some low activators are given in Table 7.3.

Long term features. We observe that some feature activations persist longer than others. We visualize some instances that activate feature 7 in the negative direction in Figure 7.5. Note how negative activation persists once there is a strong activation. This is also strengthened by strong negative activation of $(V^z\bar{h})_7$ and update gate (z_7) values that are close to zero, as seen in the plot.

We observe similar behavior when we visualize the instances that are impacted the most by the absence of feature 7. Figure 7.6, shows top two instances, where we also plot $h - h'$ next to h , where h' is the hidden representation when h_7 is forced to zero. Note how much of an impact feature 7 has on many other features, which suggests that it carries information long term and many other features use it for their own computations.

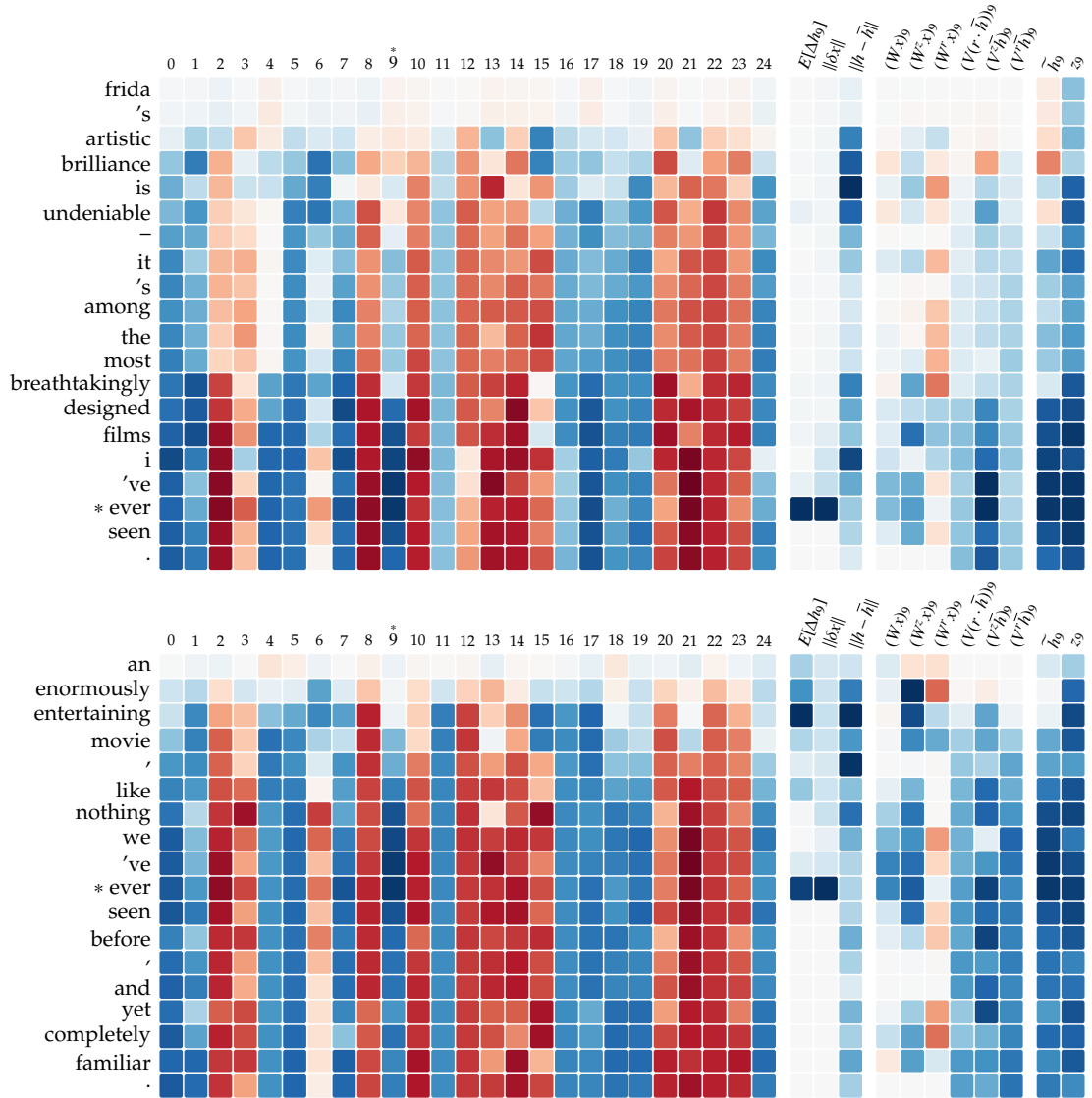


Figure 7.4: Top two activators of feature 9.

Transient features. We find that some features behave in a more “*spiky*” fashion than others, in the sense that they show frequent large changes from one timestep to the next. For instance, in many of the instances we inspect, we observe high amounts of activation change between consecutive timesteps for feature 15. Curiously, this is less apparent when looking at the highest activations of the feature, it seems that such spiky behavior occurs more frequently

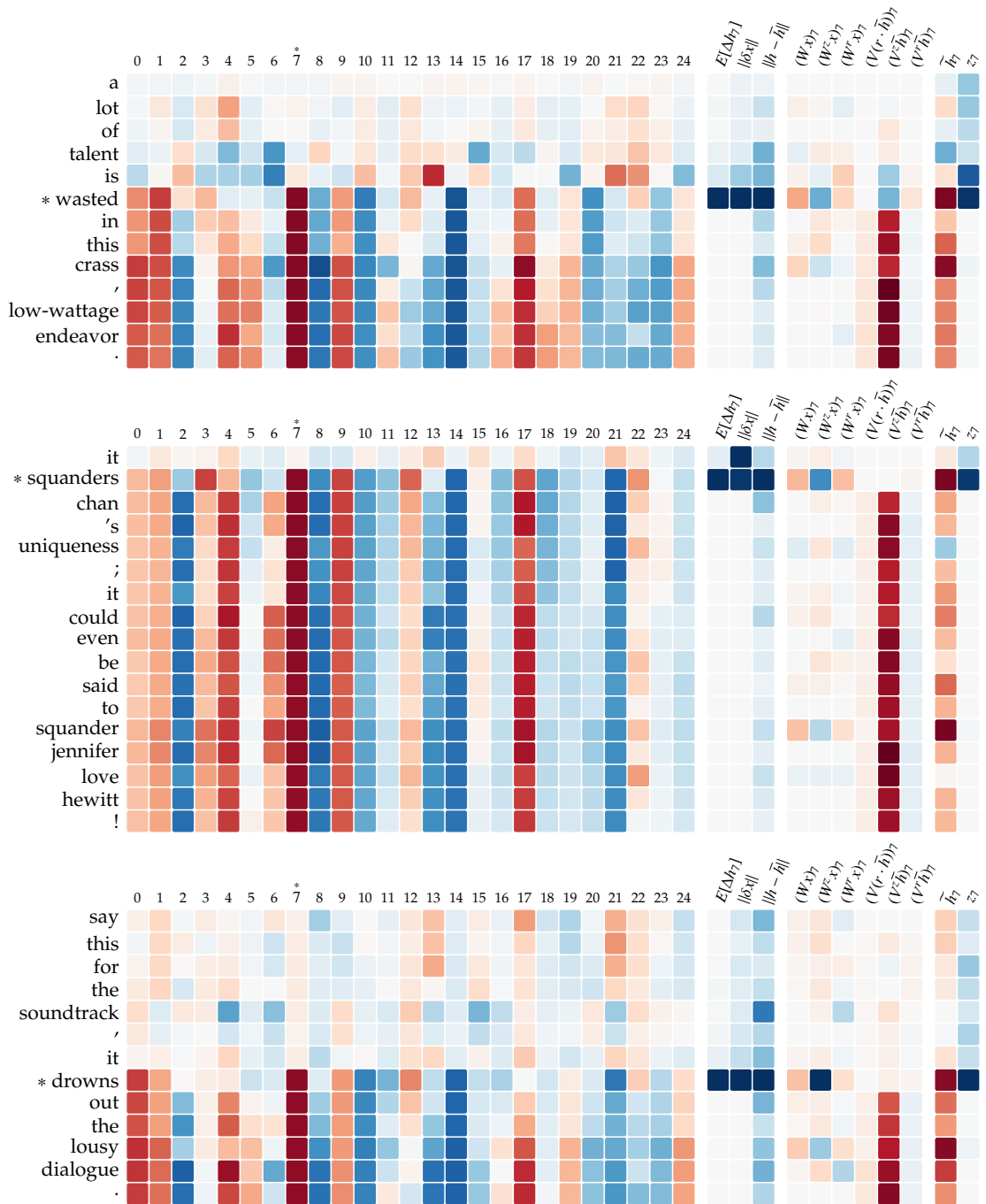


Figure 7.5: Some low activators of feature 7.

the acting is <u>amateurish</u> , the cinematography is <u>atrocious</u> , the direction is <u>clumsy</u> , the writing is <u>insipid</u> and the violence is at once <u>luridly graphic</u> and <u>laughably unconvincing</u> .
this <u>wretchedly unfunny wannabe comedy</u> is <u>inane</u> and <u>awful</u> - no doubt , it 's the <u>worst movie</u> i 've seen this summer .
the graphic carnage and re-creation of war-torn croatia is <u>uncomfortably timely</u> , <u>relevant</u> , and <u>sickeningly real</u> .
<u>excruciatingly unfunny</u> and <u>pitifully unromantic</u> .
this <u>painfully unfunny farce</u> <u>traffics in tired stereotypes</u> and <u>encumbers itself with complications</u> ... <u>that have no bearing on the story</u> .
the plot 's <u>contrivances</u> are <u>uncomfortably strained</u> .
... <u>unlikable</u> , <u>uninteresting</u> , <u>unfunny</u> , and <u>completely</u> , <u>utterly inept</u> .
the film 's <u>final hour</u> , where nearly all the previous <u>unseen material</u> <u>resides</u> , is <u>unconvincing soap opera</u> that <u>tornatore was right to cut</u> .
borstal boy represents the worst kind of filmmaking , the kind that <u>pretends to be passionate and truthful</u> but is really <u>frustratingly timid and soggy</u> .
the predominantly <u>amateur cast</u> is <u>painful to watch</u> , so <u>stilted and unconvincing</u> are the <u>performances</u> .

Table 7.3: Lowest activators of feature 9. Activations are coded with underline colors.

when the feature is away from saturation.

To quantify the notion of spikyness, for each feature we measure the mean immediate squared change (between timesteps t and $t - 1$) averaged over the entire data, shown in Figure 7.7. Indeed, we see that feature 15 has the highest value, confirming our manual observation.

Still, we find it difficult to isolate the behavior of feature 15. Highest activating instances contain words such as *watchable*, *humor*, *comedic*, *movie*, *funny*, *unfunny*, in the original dataset and the unsupervised dataset. However when we investigate the most deviated sentences when feature 15 is absent, we do not see a trivial common pattern, or a common structure shared with the activators.

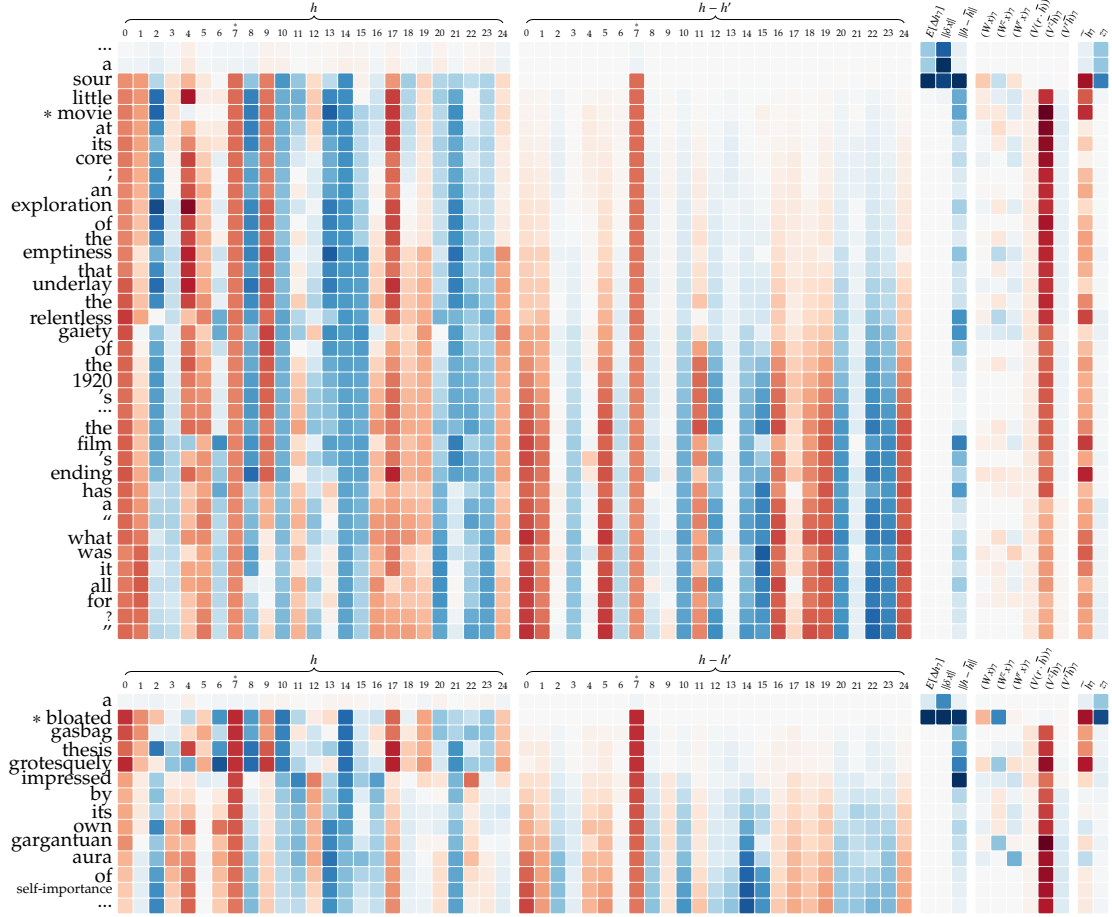


Figure 7.6: Some instances that have high output deviation with the absence of feature 7.

Saliency and expected deviation. We observe many instances on which saliency and expected deviation behave similarly and many on which they disagree. This suggests a complementary behavior between the two measures and that they provide different kind of information. One should be aware that, because of its definition, expected deviation can assign importance to a word based on what it is and *what it is not*. This sometimes can be seen in the instances where there is high expected deviation on stop-words such as “to” or “and”, where the substitute model suggests words that have impact in their place.

In Figure 7.8 we see different behavior for the two measures. In general, in

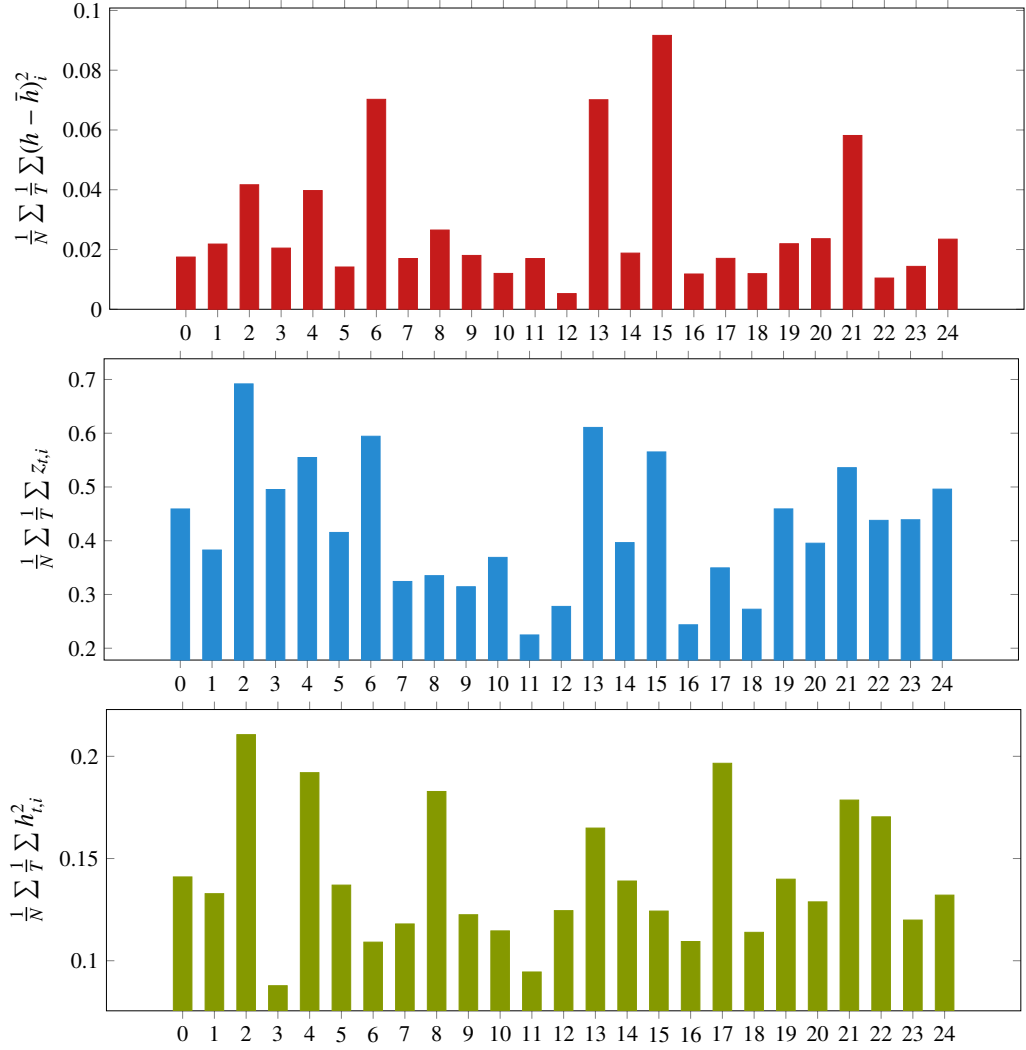


Figure 7.7: Some aggregate statistics of 25 features. **Top.** Mean average immediate squared change (spikyness). **Middle.** Mean average update gate activation. **Bottom.** Mean average squared activation (squared distance to zero).

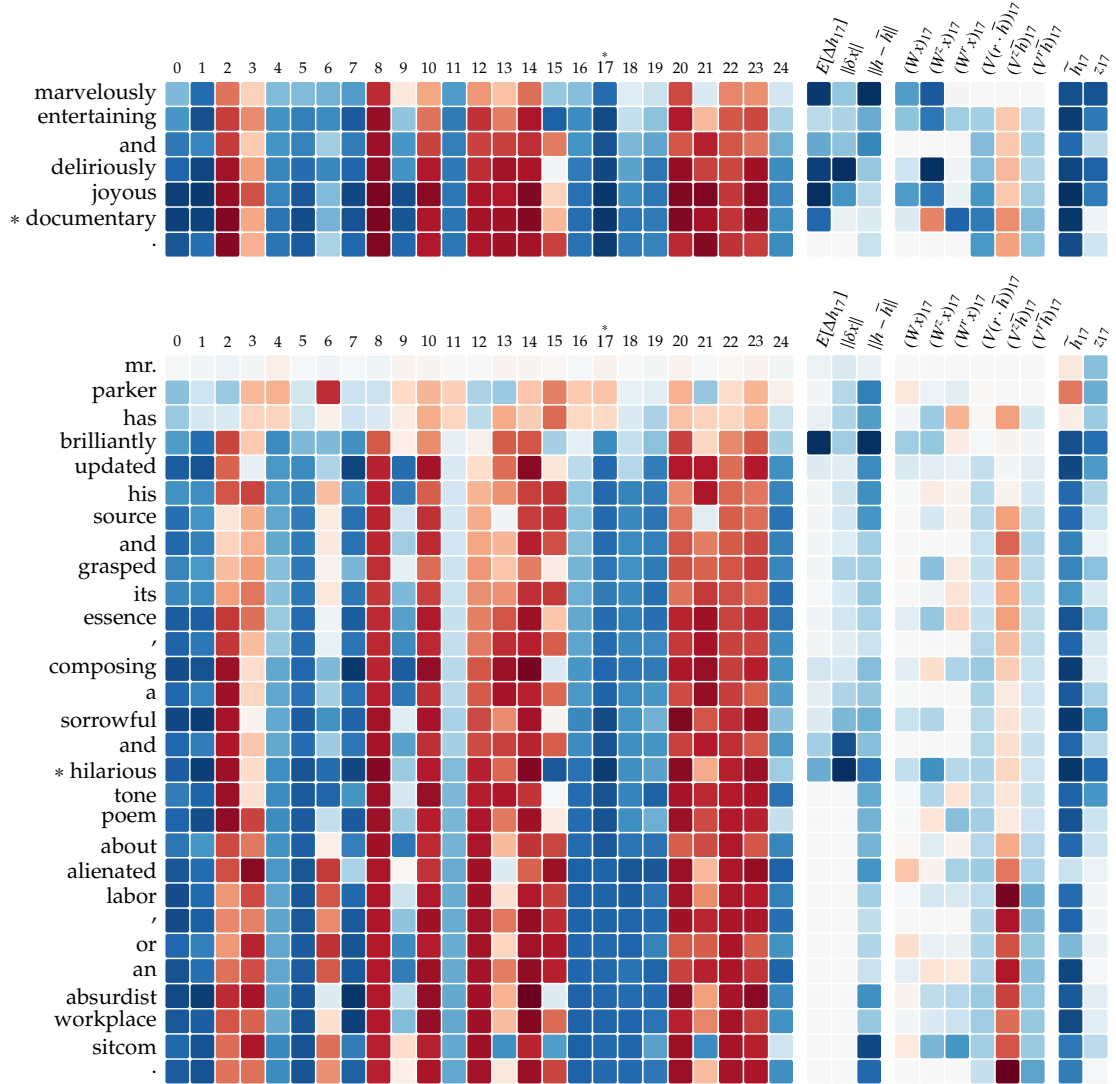


Figure 7.8: Some instances that have high activation of feature 17. Saliency and expected deviations differ.

most high activators of this particular feature (17), we see consistently different behavior where expected deviation assigns importance to words that are farther back from the timestep of interest. A similar behavior is consistently observed for feature 18 as well, which is visualized in Figure 7.9. In this case, high assignment of expected deviation also matches the timestep where fine-grained activations $(V\bar{h})_{18}$, $(V^z\bar{h})_{18}$, and $(V^r\bar{h})_{18}$ start to change, which confirms that the

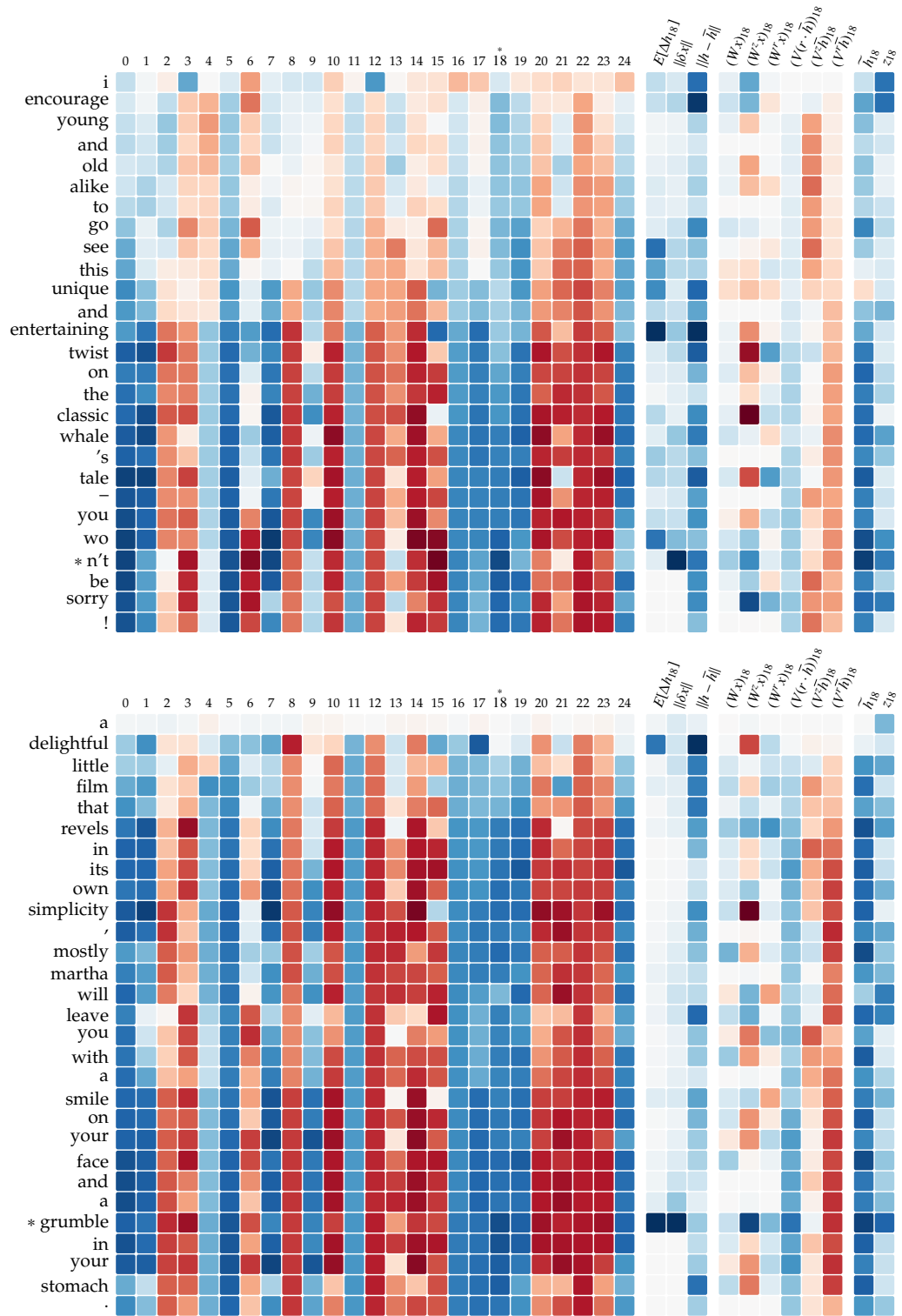


Figure 7.9: Some instances that have high activation of feature 18. Saliency and expected deviations differ.

word carries some notion of importance to the activation.

We observe the opposite behavior as well. In some of our previously mentioned results, such as Figure 7.2 on feature 4, we see that expected deviation assigns nearly all the impact to the most recent word. This suggest that long / short term behavior might not necessarily be an artifact over one metric against the other, but rather depend on the particular input.

In the case of saliency, we find that this behavior also highly depends on whether the activation of interest is saturated or not. When we investigate instances that have high deviation with the absence of feature 18, interestingly, most activations of the feature are close to zero, hence, not saturated. In those instances, saliency and expected deviation metrics seem to mostly agree.

Decorrelating features with PCA. We perform some of the activator analyses on decorrelated feature representations as well. This is done by applying PCA over all prefix feature activations, then using this transformed feature space to inspect individual features. The transformed features are rescaled so that they are in $[-1, 1]$ for visualization purposes.

Generally, transformed features seem more isolated compared to the original feature space. Table 7.4 shows top positive activators of feature 2, which mostly seem to focus on negation, such as *never* or *nothing*. Table 7.5 shows the results for feature 4, which seemingly mostly contain proper names.

this <u>beautifully</u> <u>animated</u> <u>epic</u> is <u>never</u> <u>dull</u> .
i 've <u>never</u> <u>seen</u> or <u>heard</u> <u>anything</u> <u>quite</u> <u>like</u> <u>this</u> <u>film</u> , and i <u>recommend</u> it for its <u>originality</u> <u>alone</u> .
i 've <u>never</u> <u>seen</u> -lrb- a <u>remake</u> -rrb- <u>do</u> <u>anything</u> <u>as</u> <u>stomach-turning</u> <u>as</u> <u>the</u> <u>way</u> <u>adam</u> <u>sandler</u> 's <u>new</u> <u>movie</u> <u>rapes</u> , <u>pillages</u> and <u>incinerates</u> <u>frank</u> <u>capra</u> 's <u>classic</u> ...
i 've <u>never</u> <u>bought</u> from <u>telemarketers</u> , but i <u>bought</u> <u>this</u> <u>movie</u> .
<u>hugh</u> <u>grant</u> , who has a <u>good</u> <u>line</u> in <u>charm</u> , has <u>never</u> <u>been</u> <u>more</u> <u>charming</u> <u>than</u> in about a <u>boy</u> .
director <u>clare</u> <u>kilner</u> 's <u>debut</u> is <u>never</u> <u>as</u> <u>daft</u> as it <u>should</u> <u>have</u> <u>been</u> .
<u>viveka</u> <u>seldahl</u> and <u>sven</u> <u>wollter</u> will <u>touch</u> <u>you</u> to the <u>core</u> in a <u>film</u> you will <u>never</u> <u>forget</u> – that you <u>should</u> <u>never</u> <u>forget</u> .
with a <u>tone</u> as <u>variable</u> as the <u>cinematography</u> , <u>schaeffer</u> 's <u>film</u> <u>never</u> <u>settles</u> into the <u>light-footed</u> <u>enchantment</u> the <u>material</u> <u>needs</u> , and the <u>characters</u> ' <u>quirks</u> and <u>foibles</u> <u>never</u> <u>jell</u> into <u>charm</u> .
offers absolutely <u>nothing</u> i <u>had</u> <u>n't</u> <u>already</u> <u>seen</u> .
<u>wickedly</u> <u>funny</u> , <u>visually</u> <u>engrossing</u> , <u>never</u> <u>boring</u> , <u>this</u> <u>movie</u> <u>challenges</u> <u>us</u> to <u>think</u> about the <u>ways</u> <u>we</u> <u>consume</u> <u>pop</u> <u>culture</u> .

Table 7.4: Highest activators of feature 2 after PCA. Activations are coded with underline colors.

7.5 Chapter Summary

In this chapter we discuss the interpretation of compositional neural models in natural language processing. We identify several challenges involved when inspecting unit activators in the context of NLP, due to the discrete and temporal nature of the input space. We propose simple tools to visualize feature activations and perform manual exploratory analyses on the features learned by a sequence model on the task of sentiment analysis. Our observations include:

- There are some features that are somewhat isolated around individual occurrences of words related to certain topics or concepts (such as movie keywords) however most features seem to be generic positive or negative sentiment related, and their interactions are unclear. Features that strongly

director <u>chris wedge</u> and screenwriters <u>michael berg</u> , <u>michael j. wilson</u> and <u>peter ackerman</u> create some episodes that rival vintage looney tunes for the most creative mayhem in a brief amount of time .
as written by <u>michael berg</u> and <u>michael j. wilson</u> from a story by <u>wilson</u> , this relentless , all-wise-guys-all-the-time approach tries way too hard and gets <u>tiring in no time at all</u> .
the <u>densest distillation</u> of roberts ' movies ever made .
director <u>paul cox</u> 's unorthodox , abstract approach to visualizing <u>nijinsky 's diaries</u> is both stimulating and demanding .
no matter how much he runs around and acts like a doofus , accepting a <u>50-year-old</u> in the role is creepy in a <u>michael jackson</u> sort of way .
director <u>shekhar kapur</u> and screenwriters <u>michael schiffer</u> and <u>hossein amini</u> have tried hard to modernize and reconceptualize things , but the barriers finally prove to be too great .
<u>paul cox</u> needed to show it .
<u>paul bettany</u> playing <u>malcolm mcdowell</u> ?
<u>paul bettany</u> is good at being the ultra-violent gangster wannabe , but the movie is certainly not number 1 .
<u>paul bettany</u> is cool .

Table 7.5: Highest activators of feature 4 after PCA. Activations are coded with underline colors.

respond to certain words can be explained further by looking at the activating word vectors of the word contribution at current timestep, $(Wx)_i$.

- Features have different temporal behavior in that some of them change smoothly over time and stay on for long amounts of time once activated, and others that consistently change very frequently.
- First order saliency and expected deviation can be used to assign importance to words on a feature activation of interest, and they behave in a complementary fashion. Expected deviation can be more useful when a feature is close to saturation and derivatives are near zero. Expected deviation can also show long-term dependencies where a neuron is affected by a word that comes before, without firing that neuron immediately.
- Cross-feature saliency analysis is very limited due to similar issues about

once the expectation of laughter has been quashed by whatever obscenity is at hand , even the funniest idea is n't funny .
watching this film , what we feel is n't mainly suspense or excitement .
the movie is n't painfully bad , something to be ' fully experienced ' ; it 's just tediously bad , something to be fully forgotten .
originality ai n't on the menu , but there 's never a dull moment in the giant spider invasion comic chiller .
astonishing is n't the word – neither is incompetent , incoherent or just plain crap .
schaeffer is n't in this film , which may be why it works as well as it does .
but here 's the real damn : it is n't funny , either .
you really have to salute writer-director haneke -lrb- he adapted elfriede jelinek 's novel -rrb- for making a film that is n't nearly as graphic but much more powerful , brutally shocking and difficult to watch .
a sharp and quick documentary that is funny and pithy , while illuminating an era of theatrical comedy that , while past , really is n't .
a sensitive and expertly acted crowd-pleaser that is n't above a little broad comedy and a few unabashedly sentimental tears .

Table 7.6: Top negative activators of feature 7 after PCA. Activations are coded with underline colors.

derivatives being close to zero. On the other hand, clamping individual features to zero and measuring the impact on every other feature yields more promising results. We see that certain long term features (e.g. feature 7) have very strong and lasting impact on the overall memory representation in general.

- Measuring activations at a finer-grained level, such as looking at update gates z , candidate memory \tilde{h} , or individual contributions of input x or past \tilde{h} to the current memory computation in GRUs, can provide additional insights about when the memory tends to update by adding new information from the current word, or when it votes to shut down the update gates and retain older information.
- Decorrelating the feature space using PCA seems to result in features that are more cleanly separated, and sometimes more isolated on easily inter-

pretable topics or concepts, such as negation, names or other terms. However highest variance directions (first few features after PCA) still seem to be about generic positive or negative sentiment without an obvious finer-grained distinction.

Our results in this chapter are not conclusive. In particular, to make manual inspection tractable, we perform analyses over a single model applied to a relatively simple task. However we believe that the insights and observations presented here can lay a foundation for further exploration along this front, allow us to formulate concrete hypotheses about individual features and test them, and allow us to transfer some of the insights to other applications.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In this dissertation, we presented compositional neural models for representation learning in natural language processing. The main strengths of such models are that they learn features themselves as opposed to models that require manual feature engineering and they explicitly address how individual words are composed into larger units of text.

8.1 Summary of Contributions

In Chapter 3 we presented an application of deep recurrent neural networks to the task of opinion mining. We achieved new state-of-the-art on the task, outperforming CRF-based approaches that rely on engineered features, and demonstrating that effective feature learning is possible for opinion extraction.

Chapter 4 presented applications of multiplicative recurrent neural networks to the task of sentiment analysis. We showed that multiplicative recurrent networks perform comparably or better than their regular Elman-type counterparts. Furthermore, they outperform matrix-space models and do so using a much smaller number of parameters. Additionally, we show how the multiplicative network is actually a simplification of the matrix-space model by parametrizing every word matrix as a multiplication of an individual vector and global third-order tensor.

Chapter 5 proposed the bidirectional recursive neural network to incorporate structure information (e.g. given by a parse tree) when performing token-level labeling tasks. Specifically, we again apply the model to the task of opinion

mining and show that it can improve accuracy over purely sequential models such as recurrent networks.

Chapter 6 proposed the deep recursive neural network, to incorporate the notion of *depth in space* to recursive computation, and implement feature hierarchies. Deep recursive networks are constructed by stacking multiple recursive layers. We applied the model to fine-grained sentiment classification and achieved a state-of-the-art, showcasing the strength of having this extra notion of depth in recursive compositional models.

Finally, Chapter 7 focused on the challenges of interpreting neural compositional models in the specific context of NLP. We have discussed issues that are specific to the nature of natural language applications such as discreteness and having a temporal dimension. We proposed several tools following up on previous work to visualize unit activations and perform manual inspections of individual features.

8.2 Future Work

Our work in this dissertation has demonstrated that compositional neural models can be effectively used in natural language problems to learn representations that respect compositionality in language without manual feature engineering and domain expertise. In the following, we discuss future work in several directions:

Towards multitask learning and general natural language understanding. All of the compositional models we have applied throughout this dissertation

were task specific. In contrast, multitask learning in NLP has been of interest in previous work. Since common concepts in language would apply to individual tasks, it is intuitive to share information across tasks. This could be seen as the first step towards a general, task-independent natural language understanding model. Even though there has been interest in multitask learning specifically with neural models, improvements remain relatively small and the best mechanism for knowledge sharing across tasks is unclear.

Semi-supervised and unsupervised learning of general compositional models. Another common aspect of all our experiments was that they were all performed in the supervised setting. Word vector representations can be seen as a proof of effective unsupervised learning in natural language, which should encourage researchers to find better methods of unsupervised learning beyond word level. Previous work such as paragraph vectors (Le and Mikolov, 2014a), sequential autoencoders (Luong et al., 2015; Li et al., 2015b) or skip-thought vectors (Kiros et al., 2015) can be seen as steps towards this end. Considering the initial impact of word vector representations on a wide variety of tasks, such an unsupervised compositional model can similarly greatly improve the performance on many tasks, especially ones that have limited availability of labeled data. This would also help the above goal towards better multitask learning.

Better model interpretation. In Chapter 7 we addressed the challenges we have described in a very limited fashion. To make manual inspection easy, we looked into a single model on a single application over sentiment classification. Therefore, an obvious possible direction we intend to explore is experimenting with different models on different tasks, to see if there are representations that are overlapping across tasks, and representations that are specifically tailored

towards a given task.

Another limitation of our analysis was that it uses a single pretrained word-vector lookup table. We expect model behavior to depend highly on the initial word representation being used, therefore a relevant direction we intend to pursue is to explore this dependence. For instance, dependency tree based training of word vectors are known to induce representations that put more emphasis into syntax than semantics (Levy and Goldberg, 2014c). Hence, features induced by a compositional model built on top of these word vectors might include better or finer-grained syntax information, compared to skip-gram vectors.

A possible orthogonal direction we would like to explore is to quantify feature representations in a more concrete and formal manner, perhaps by using interpretable models, such as rule-based learners, to mimic the behavior of individual feature representations to explain those features. One could also hand design a rule based model based on a hypothesis about a given feature, and then test its predictive power on the feature activations as a way of testing the hypothesis.

APPENDIX A

ADDITIONAL VISUALIZATIONS OF MODEL ACTIVATIONS

A.1 Cross-feature Saliency

We have generally found cross-feature saliency visualizations less informative and omitted our results from Chapter 7. Main reason for this is that in most cases, many of the derivatives become very close to zero (see Figure A.1). We have observed some positive, possibly more informative instances as well, where derivatives are not as close to zero (see Figure A.2).

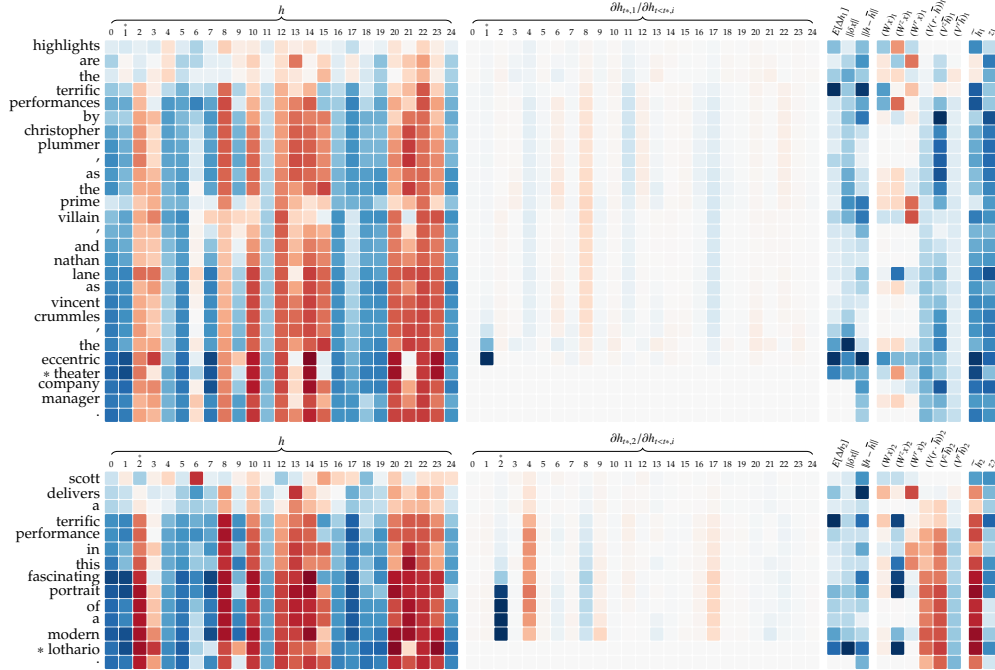


Figure A.1: Some activations visualized with cross-feature saliency values (second block of columns). Most values are close to zero.

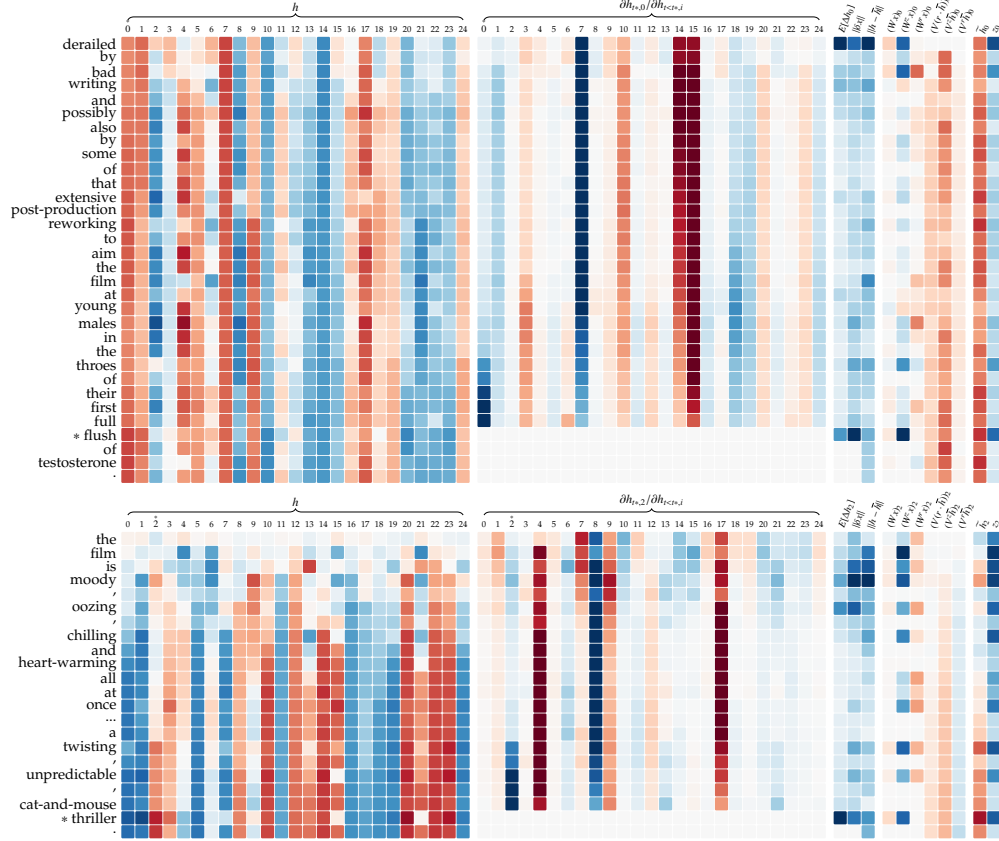


Figure A.2: More activations visualized with cross-feature saliency values (second block of columns). Values are away from zero.

A.2 Generic Sentiment Features

Many of the original features learned by the GRU network seem to be generic positive or negative sentiment with little to no obvious isolation in terms of a topic or concept. To illustrate this we show activators of feature 5 and 14, in Tables A.1 and A.2, respectively.

an exhilarating futuristic thriller-noir , minority report twists the best of technology around a gripping story , delivering a riveting , pulse intensifying escapist adventure of the first order
it 's a treat – a delightful , witty , improbable romantic comedy with a zippy jazzy score ... grant and bullock make it look as though they are having so much fun .
... the first 23 of the film are incredibly captivating and insanely funny , thanks in part to interesting cinematic devices -lrb- cool visual backmasking -rrb- , a solid cast , and some wickedly sick and twisted humor ...
unexpected moments of authentically impulsive humor are the hallmark of this bittersweet , uncommonly sincere movie that portrays the frank humanity of ... emotional recovery .
wickedly funny , visually engrossing , never boring , this movie challenges us to think about the ways we consume pop culture .
jae-eun jeong 's take care of my cat brings a beguiling freshness to a coming-of-age story with such a buoyant , expressive flow of images that it emerges as another key contribution to the flowering of the south korean cinema .
... a delightfully unpredictable , hilarious comedy with wonderful performances that tug at your heart in ways that utterly transcend gender labels .

Table A.1: Top activators of feature 5. Activations are coded with underline colors.

A.3 Decorrelated Features

After applying PCA to the feature space, heatmap plots are visibly less correlated (see Figure A.3). Perhaps as expected, first two features (two directions that have highest variance) seem to behave like generic sentiment detectors.

Next set of features seem more and more isolated around a concept, or a cluster of words, as seen in Chapter 7. We present additional set of activators in Tables A.3, A.4, and A.5.

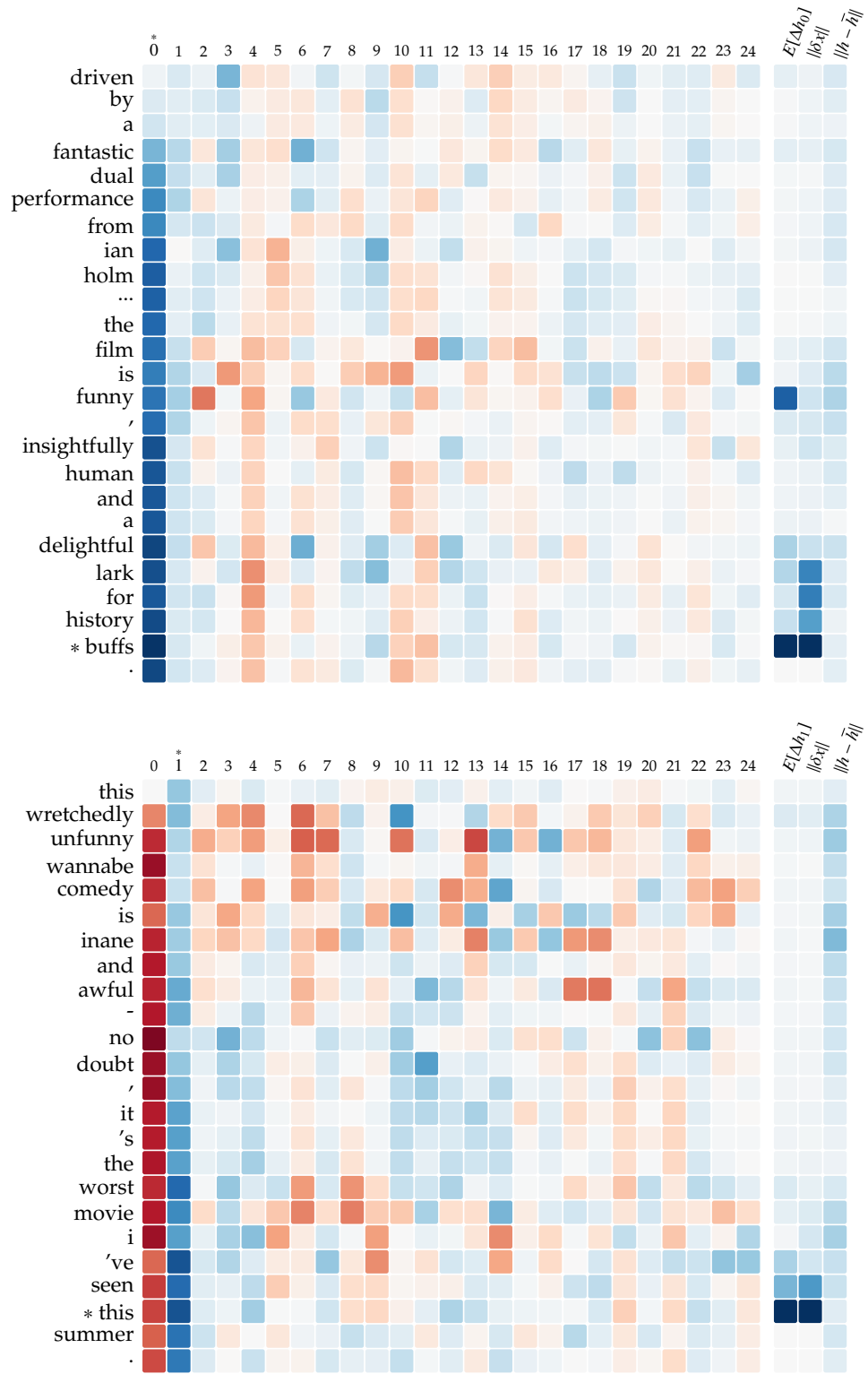


Figure A.3: Activators of decorrelated feature 0 (top) and 1 (bottom)

the acting is amateurish , the cinematography is atrocious , the direction is clumsy , the writing is insipid and the violence is at once luridly graphic and laughably unconvincing .

this overproduced piece of dreck is shockingly bad and absolutely unnecessary .

mandel holland 's direction is uninspired , and his scripting unsurprising , but the performances by phifer and black are ultimately winning .

a thoroughly awful movie – dumb , narratively chaotic , visually sloppy ... a weird amalgam of ' the thing ' and a geriatric ' scream . '

... unlikable , uninteresting , unfunny , and completely , utterly inept .

the premise itself is just soooooo tired .

the satire is unfocused , while the story goes nowhere .

hilariously inept and ridiculous .

a beautifully made piece of unwatchable drivel .

horrendously amateurish filmmaking that is plainly dull and visually ugly when it is n't incomprehensible .

Table A.2: Top activators of feature 14. Activations are coded with underline colors.

try this obscenely bad dark comedy , so crass that it makes edward burns ' sidewalks of new york look like oscar wilde .

there is one surefire way to get a nomination for a best-foreign-film oscar : make a movie about whimsical folk who learn a nonchallenging , life-affirming lesson while walking around a foreign city with stunning architecture .

ford deserves to be remembered at oscar time for crafting this wonderful portrait of a conflicted soldier .

despite the surface attractions – conrad l. hall 's cinematography will likely be nominated for an oscar next year – there 's something impressive and yet lacking about everything .

oscar wilde 's masterpiece , the importance of being earnest , may be the best play of the 19th century .

oscar caliber cast does n't live up to material

gooding and coburn are both oscar winners , a fact which , as you watch them clumsily mugging their way through snow dogs , seems inconceivable .

australian actordirector john polson and award-winning english cinematographer giles nuttgens make a terrific effort at disguising the obvious with energy and innovation .

Table A.3: Top negative activators of feature 5 after PCA. Activations are coded with underline colors.

a gracious , eloquent film that by its end offers a ray of hope to the refugees able to look ahead and resist living in a past forever lost .

were dylan thomas alive to witness first-time director ethan hawke 's strained chelsea walls , he might have been tempted to change his landmark poem to , ' do not go gentle into that good theatre . '

a stirring tribute to the bravery and dedication of the world 's reporters who willingly walk into the nightmare of war not only to record the events for posterity , but to help us clearly see the world of our making .

this documentary is a dazzling , remarkably unpretentious reminder of what -lrb- evans -rrb- had , lost , and got back .

here is a divine monument to a single man 's struggle to regain his life , his dignity and his music .

an engrossing iranian film about two itinerant teachers and some lost and desolate people they encounter in a place where war has savaged the lives and liberties of the poor and the dispossessed .

a thoughtful and surprisingly affecting portrait of a screwed-up man who dared to mess with some powerful people , seen through the eyes of the idealistic kid who chooses to champion his ultimately losing cause .

Table A.4: Top activators of feature 7 after PCA. Activations are coded with underline colors.

excellent performances from jacqueline bisset and martha plimpton grace this deeply touching melodrama .

brian tufano 's handsome widescreen photography and paul grabowsky 's excellent music turn this fairly parochial melodrama into something really rather special .

gooding is the energetic frontman , and it 's hard to resist his enthusiasm , even if the filmmakers come up with nothing original in the way of slapstick sequences .

pure cinematic intoxication , a wildly inventive mixture of comedy and melodrama , tastelessness and swooning elegance .

a gem of a romantic crime comedy that turns out to be clever , amusing and unpredictable .

cute , funny , heartwarming digitally animated feature film with plenty of slapstick humor for the kids , lots of in-jokes for the adults and heart enough for everyone .

a truly wonderful tale combined with stunning animation .

aside from being the funniest movie of the year , simone , andrew niccol 's brilliant anti-hollywood satire , has a wickedly eccentric enchantment to it .

wonderful fencing scenes and an exciting plot make this an eminently engrossing film .

Table A.5: Top activators of feature 10 after PCA. Activations are coded with underline colors.

BIBLIOGRAPHY

- Heike Adel, Ngoc Thang Vu, and Tanja Schultz. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *ACL (2)*, pages 206–211, 2013.
- Michael Auli and Jianfeng Gao. Decoder integration and expected bleu training for recurrent neural network language models. In *ACL (2)*, pages 136–142, 2014.
- Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054. Association for Computational Linguistics, 2013. URL <http://aclweb.org/anthology/D13-1106>.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- Marco Baroni and Roberto Zamparelli. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics, 2010.
- Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.
- Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

- Yoshua Bengio, Rjean Ducharme, Pascal Vincent, Christian Jauvin, Jaz K, Thomas Hofmann, Tomaso Poggio, and John Shawe-taylor. A neural probabilistic language model. In *In Advances in Neural Information Processing Systems*, 2001.
- Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
- Eric Breck, Yejin Choi, and Claire Cardie. Identifying expressions of opinion in context. In *IJCAI*, pages 2683–2688, 2007.
- Jianlin Cheng, Zheng Wang, and Gianluca Pollastri. A neural network approach to ordinal regression. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1279–1284. IEEE, 2008.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/D14-1179>.
- Yejin Choi and Claire Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 793–801. Association for Computational Linguistics, 2008.

- Yejin Choi and Claire Cardie. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 269–274. Association for Computational Linguistics, 2010.
- Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of HLT/EMNLP*, pages 355–362. Association for Computational Linguistics, 2005.
- Grzegorz Chrupała. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 680–686. Citeseer, 2014.
- Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204*, 2016.
- Stephen Clark. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, 2008.
- B. Coecke, M. Sadrzadeh, and Clark S. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384, 2010.
- Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from

- scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011. ISSN 1532-4435.
URL <http://dl.acm.org/citation.cfm?id=1953048.2078186>.
- George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.
- Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603. IEEE, 2013.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.
- Kevin Duh, Graham Neubig, Katsuhito Sudoh, and Hajime Tsukada. Adaptation data selection using neural language models: Experiments in machine translation. In *ACL (2)*, pages 678–683, 2013.
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*, 2015.
- Michael Egmont-Petersen, Dick de Ridder, and Heinz Handels. Image processing with neural networks a review. *Pattern recognition*, 35(10):2279–2301, 2002.
- Salah El Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*, pages 493–499, 1995.

- Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
- Katrin Erk. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653, 2012.
- Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54, 2008.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*, volume 15, pages 315–323, 2011.
- Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.
- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772, 2014.
- Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.
- Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

- E. Grefenstette, G. Dinu, Y. Zhang, M. Sadrzadeh, and M. Baroni. Multi-step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 131–142, Potsdam, Germany, March 2013. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W13-0112>.
- Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069*, 2015.
- Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.
- Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.
- Karl Moritz Hermann and Phil Blunsom. The role of syntax in vector space models of compositional semantics. In *ACL (1)*, pages 894–904. Citeseer, 2013.
- Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 190–198, 2013.
- Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012a.

- Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012b.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- Ozan İrsoy and Claire Cardie. Bidirectional recursive neural networks for token-level labeling with structure. In *NIPS Deep Learning Workshop*, 2013.
- Ozan İrsoy and Claire Cardie. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104, 2014a.
- Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728, 2014b.
- Ozan İrsoy and Claire Cardie. Modeling compositionality with multiplicative recurrent neural networks. In *ICLR*, 2015.
- Mohit Iyyer, Jordan L Boyd-Graber, Leonardo Max Batista Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *EMNLP*, pages 633–644, 2014a.
- Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. Political ideology detection using recursive neural networks. In *Proceedings of the Association for Computational Linguistics*, 2014b.
- Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*, 2015.

Richard Johansson and Alessandro Moschitti. Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 67–76. Association for Computational Linguistics, 2010.

Richard Johansson and Alessandro Moschitti. Extracting opinion expressions and their polarities: exploration of pipelines and joint models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 101–106. Association for Computational Linguistics, 2011.

Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 2342–2350, 2015.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014. URL <http://goo.gl/EsQCuC>.

Ken-ichi Kamijo and Tetsuji Tanigawa. Stock price pattern recognition-a recurrent neural network approach. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 215–221. IEEE, 1990.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Alistair Kennedy and Diana Inkpen. Sentiment classification of movie reviews

- using contextual valence shifters. *Computational Intelligence*, 22(2):110–125, 2006.
- Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.
- Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- Phong Le and Willem Zuidema. The inside-outside recursive neural network model for dependency parsing. In *EMNLP*, pages 729–739, 2014.
- Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014a.
- Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 14, pages 1188–1196, 2014b.
- Quoc V. Le, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *International Conference on Machine Learning*, 2012.
- Rémi Lebret and Ronan Collobert. Word emdeddings through hellinger pca. *arXiv preprint arXiv:1312.5542*, 2013.

- Omer Levy and Yoav Goldberg. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, Ann Arbor, Michigan, June 2014a. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/W/W14/W14-1618>.
- Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014b.
- Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL (2)*, pages 302–308, 2014c.
- Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.
- Jiwei Li, Rumeng Li, and Eduard H Hovy. Recursive deep models for discourse parsing. In *EMNLP*, pages 2061–2069, 2014.
- Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015a.
- Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015b.
- Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*, 2015c.
- Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. Finding function in form:

- Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*, 2015.
- Jingjing Liu and Stephanie Seneff. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 161–169. Association for Computational Linguistics, 2009.
- Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.
- Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113, 2013.
- Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.
- Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.
- Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, 2013.
- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

- Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013b.
- Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *ACL*, pages 236–244, 2008.
- Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439, 2010.
- Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM, 2007.
- M Arthur Munson, Claire Cardie, and Rich Caruana. Optimizing to arbitrary nlp metrics using ensemble selection. In *Proceedings of HLT/EMNLP*, pages 539–546. Association for Computational Linguistics, 2005.
- Kouichi Murakami and Hitomi Taguchi. Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 237–242. ACM, 1991.
- Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language*

- Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics, 2010.
- Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011.
- Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436. IEEE, 2015.
- Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.
- Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, 2002.
- Razvan Pascanu, Çağlar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.
- Petra Perner. How to interpret decision trees? In *Industrial Conference on Data Mining*, pages 40–55. Springer, 2011.
- Livia Polanyi and Annie Zaenen. Contextual valence shifters. In *Computing attitude and affect in text: Theory and applications*, pages 1–10. Springer, 2006.

- J. B. Pollack. Recursive distributed representations. *Artificial Intelligence*, 1:77–105, 1990.
- Sebastian Rudolph and Eugenie Giesbrecht. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 907–916. Association for Computational Linguistics, 2010.
- Haşim Sak, Andrew Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk. Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4280–4284. IEEE, 2015.
- Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.
- Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.
- Mostafa Al Masum Shaikh, Helmut Prendinger, and Ishizuka Mitsuru. Assessing sentiment of text by semantic dependency and contextual valence analysis. In *Affective Computing and Intelligent Interaction*, pages 191–202. Springer, 2007.
- Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809, 2011a.

- Richard Socher, Cliff C Lin, Andrew Ng, and Chris Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136, 2011b.
- Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011c.
- Richard Socher, Yoshua Bengio, and Christopher D. Manning. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012, ACL '12*, pages 5–5, Stroudsburg, PA, USA, 2012a. Association for Computational Linguistics. URL <http://dl.acm.org/citation.cfm?id=2390500.2390505>.
- Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012b.
- Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '13*, 2013.
- Andreas Stolcke. SRILM – An extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November 2002.

- Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.
- Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.
- Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Sys-*

- tems* 28, pages 2773–2781. Curran Associates, Inc., 2015. URL <http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf>.
- Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. Predicting polarities of tweets by composing word embeddings with long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1343–1353, 2015.
- Taro Watanabe and Eiichiro Sumita. Transition-based neural constituent parsing. *Proceedings of ACL-IJCNLP*, 2015.
- Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- Dominic Widdows. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075114. URL <http://www.aclweb.org/anthology/P03-1018>.
- Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3): 165–210, 2005.
- Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP*, pages 347–354. Association for Computational Linguistics, 2005.

Wenduan Xu, Michael Auli, and Stephen Clark. Ccg supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 250–255, 2015.

Bishan Yang and Claire Cardie. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. Association for Computational Linguistics, 2012.

Bishan Yang and Claire Cardie. Joint inference for fine-grained opinion extraction. In *Proceedings of ACL*, 2013.

Ainur Yessenalina and Claire Cardie. Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 172–182. Association for Computational Linguistics, 2011.

Deniz Yuret. FASTSUBS: An Efficient and Exact Procedure for Finding the Most Likely Lexical Substitutes Based on an N-Gram Language Model. *Signal Processing Letters, IEEE*, 19(11):725–728, Nov 2012. doi: 10.1109/LSP.2012.2215587.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1263–1271, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL <http://www.aclweb.org/anthology/C10-1142>.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

Chenxi Zhu, Xipeng Qiu, Xinchu Chen, and Xuanjing Huang. A re-ranking model for dependency parser with recursive convolutional neural network. *arXiv preprint arXiv:1505.05667*, 2015a.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*, 2015b.